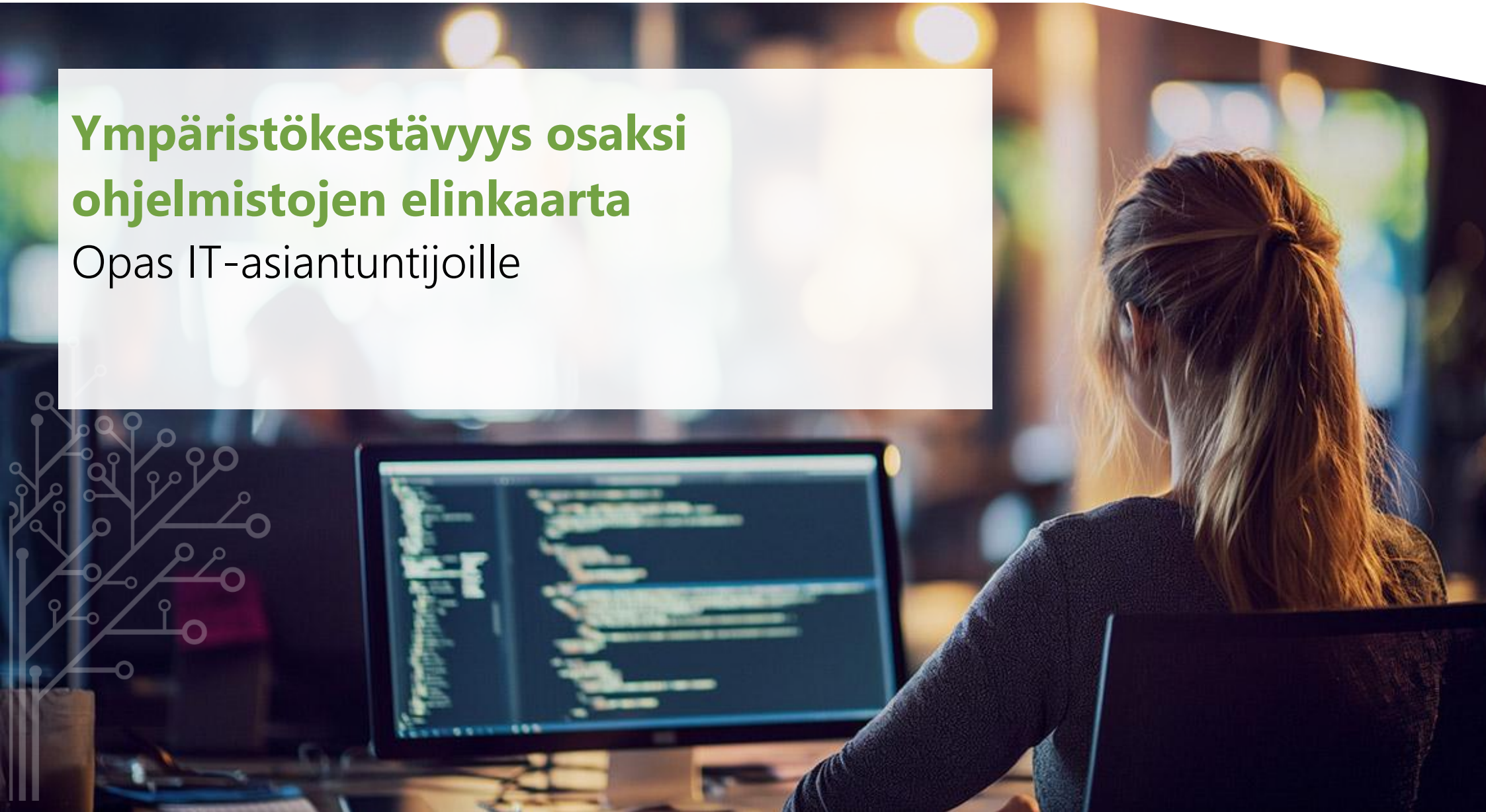
VISIIRI.

Vihreän siirtymän ICT-ekosysteemi

Euroopan unionin osarahoittama

Elinkeino-, liikenne- ja ympäristökeskus



Ympäristökestävyys osaksi ohjelmistojen elinkaarta

Opas IT-asiantuntijoille

Alkusanat

Kestävyydestä on tullut erottamaton osa yritysten strategioita, viestintää ja raportointia. Kestävyysraportit, tavoitteet ja arvot ovat näkyvästi esillä erityisesti suurten organisaatioiden verkkosivuilla. Silti yksi keskeinen kysymys jää usein vaille vastausta: miten kestävyys näkyy arjen tekemisessä, siellä missä vaikutukset lopulta syntyvät?

Ohjelmistokehitys on yksi nyky-yhteiskunnan keskeisimmistä mahdollistajista, mutta samalla myös merkittävä, usein näkymättömäksi jäävä ympäristövaikutusten lähde. Kuten ICT-alan toimijoiden kanssa käydyt keskustelut ja työpajat osoittavat, kestävyystavoitteet eivät vielä järjestelmällisesti jalkaudu ohjelmistokehityksen tasolle. Vastuu jää helposti yksittäisille kehittäjille ilman selkeitä tavoitteita, mittareita tai yhteistä ymmärrystä siitä, mitä kestävyys käytännössä tarkoittaa koodissa, arkkitehtuurissa ja teknologiavalinnoissa.

Tämä opas on syntynyt todellisesta tarpeesta kuroa umpeen kuilua strategisten kestävyystavoitteiden ja käytännön ohjelmistokehityksen välillä. Sen tarkoituksena on tehdä kestävyydestä konkreettista, ymmärrettävää ja toteutettavaa ohjelmistokehityksen arjessa, ei vain ohjeina, vaan osana päätöksentekoa, vastuunjakoa ja ammattitaitoa. Kestävä ohjelmistokehitys ei ole yksittäinen tekninen optimointi tai ylimääräinen vaatimus, vaan tapa ajatella, suunnitella ja toimia.

Todellinen vaikutus syntyy vasta silloin, kun kestävyys ei jää analyysiksi tai raportoinniksi, vaan muuttuu teoiksi. Toivomme, että tämä opas tukee juuri tätä muutosta.

Jari Porras
Professori, LUT School of Engineering Sciences
VISIIRI-hanke

Sisällysluettelo

Tiivistelmä

Julkaisun nimi
Ympäristökestävyys osaksi ohjelmistojen elinkaarta: Opas IT-asiantuntijoille

Oppaan on koonnut
LUT-yliopisto, Ohjelmistotuotannon laitos

Julkaisuvuosi
2026

Kestävyys ei synny yksittäisistä teknisistä optimoinneista, vaan siitä, miten ohjelmistoja suunnitellaan, rakennetaan, käytetään ja ylläpidetään osana laajempaa toimintaa. Tässä oppaassa tarkastellaan, miten ympäristökestävyys voidaan tuoda osaksi ohjelmistokehitystä ja IT-asiantuntijatyötä käytännönläheisesti ja realistisesti.

Kestävyys jää IT- ja ohjelmistotyössä usein abstraktiksi tavoitteeksi ja vastuu valuu yksittäisille kehittäjille ilman yhteisiä pelisääntöjä, mittareita tai päätöksentekoa tukevia rakenteita. Samalla teknologiat, kuten pilvipalvelut ja tekoäly, yleistyvät nopeasti ja muuttavat työn arkea. Tarvitaan yhteistä ymmärrystä siitä, mitä kestävyys tarkoittaa ohjelmistojen elinkaaren eri vaiheissa ja millaisia valintoja se edellyttää.

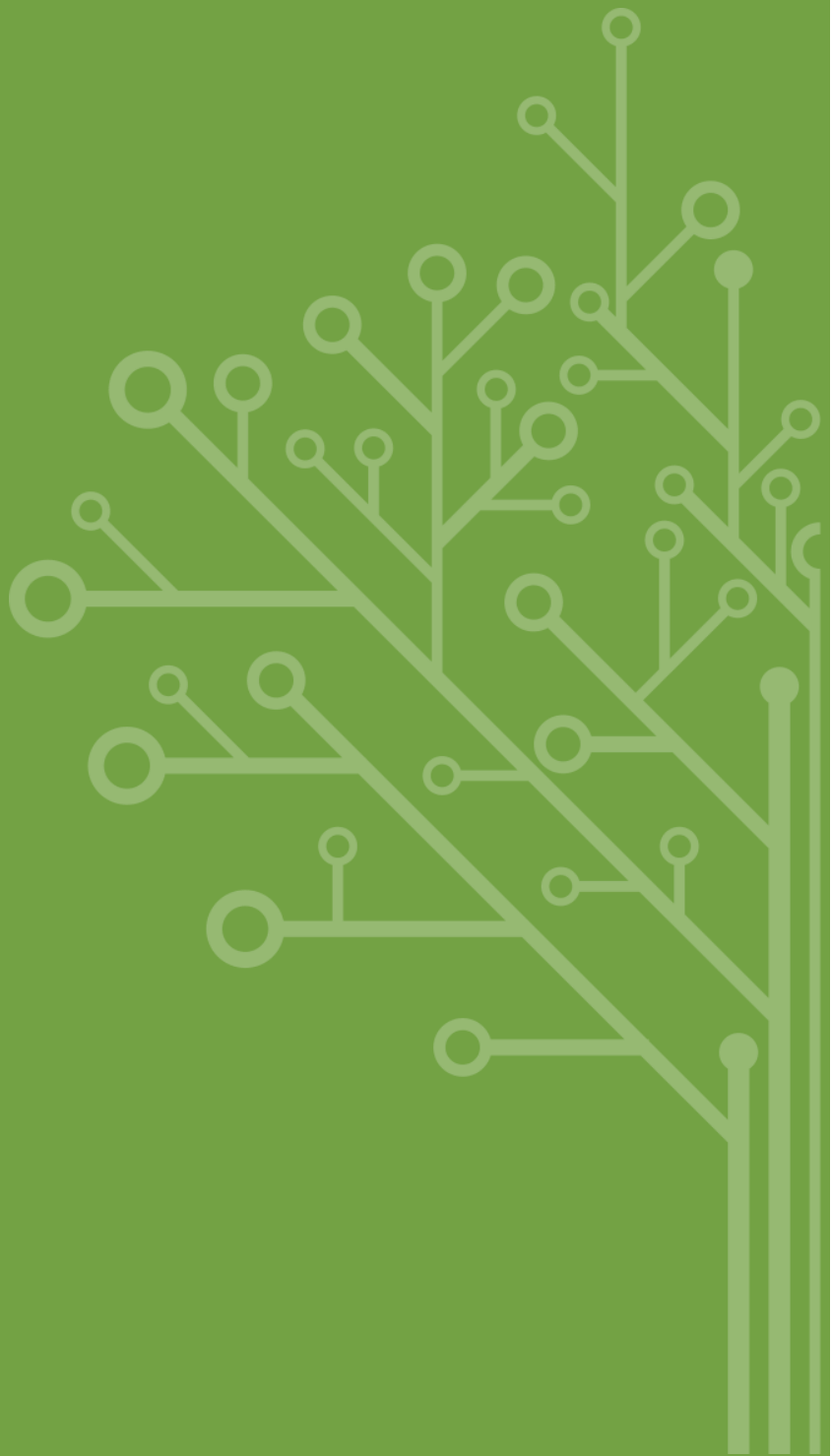
Oppaassa käsitellään ohjelmistojen ympäristövaikutuksia koko elinkaaren näkökulmasta: vaatimusten määrittelystä ja hankinnoista arkkitehtuuriin, kehitykseen, testaukseen, käyttöön ja lopulta käytöstä poistoon. Keskiössä ovat erityisesti energiatehokkuus, datan ja laskennan tarpeellisuus, pilvi- ja infrastruktuurivalinnat sekä päätöksenteon läpinäkyvyys. Opas korostaa, että kestävyyttä ei saavuteta vain mittaamalla, vaan ennen kaikkea karsimalla turhaa, tekemällä tietoisia valintoja ja rakentamalla kestävyys osaksi arjen työkäytäntöjä.

Opas on suunnattu ohjelmistokehityksen ja IT:n parissa työskenteleville asiantuntijoille, tiimeille ja organisaatioille, jotka haluavat ymmärtää, miten ympäristökestävyys voidaan huomioida osana normaalia tekemistä ilman että työn sujuvuus tai laatu kärsii.

Avainsanat – Kestävä ohjelmistokehitys, vihreä IT, ympäristökestävyys, ohjelmistojen elinkaari, energiatehokkuus, data ja laskenta, pilvipalvelut, tekoäly, GreenOps, FinOps, IT-arkkitehtuuri, hankinnat

1

Miksi kestävä IT on tärkeää?



VISIIRI.

Euroopan unionin osarahoittama

Miksi kestävä IT on tärkeää?

IT-ala on muodostunut nykyaikaisen liiketoiminnan korvaamattomaksi tukipilariksi, joka vauhdittaa innovaatioita, tuottavuutta ja globaalia yhteen liittyneisyyttä. Lukuisista hyödyistään huolimatta IT-infrastruktuuri ja digitaaliset palvelut aiheuttavat merkittäviä ympäristöhaasteita, erityisesti kasvihuonekaasupäästöjen muodossa. Useiden arvioiden mukaan tieto- ja viestintäteknologian (ICT) osuus maailmanlaajuisista päästöistä on useita prosentteja [1-4] ja ympäristöjalanjäljen odotetaan kasvavan edelleen dataintensiivisten teknologioiden, kuten tekoälyn (AI), big datan analytiikan ja esineiden internetin (IoT), yleistymisen myötä [5]. Tästä syystä organisaatioiden on yhä kiireellisempää ottaa käyttöön kestäviä IT-käytäntöjä ympäristövaikutusten hillitsemiseksi.

Ympäristövastuun lisäksi kestävä IT tarjoaa myös konkreettisia taloudellisia hyötyjä. Tehostamalla energiatehokkuutta laitteistoissa, ohjelmistoissa ja konesaleissa organisaatiot voivat merkittävästi pienentää käyttökustannuksiaan. Yhdysvaltain ympäristönsuojeluviraston (EPA) ENERGY STAR -ohjeistusten mukaan konesalit voivat saavuttaa huomattavia energiansäästöjä muun muassa kuormanhallinnan, jäähdytyksen optimoinnin ja energiatehokkaiden laitteiden avulla [6]. Lisäksi virtualisointi, energiatehokas ohjelmistosuunnittelu ja resurssien älykäs käyttö voivat pidentää IT-omaisuuden elinkaarta ja parantaa järjestelmien luotettavuutta, mikä tuottaa suoraa liiketoiminnallista lisäarvoa [7].

Myös sääntelykehykset kehittyvät vastaamaan kestävyyshaasteeseen. Hallitukset ja kansainväliset toimijat ottavat käyttöön yhä tiukempia raportointivaatimuksia ja ympäristöstandardeja yritystoiminnalle, mukaan lukien digitaaliset prosessit. Euroopan unionin Corporate Sustainability Reporting Directive (CSRD) velvoittaa yrityksiä raportoimaan ympäristö- ja sosiaalisista vaikutuksistaan, ja direktiivin virallinen oikeudellinen teksti on julkaistu EUR-Lexissä [8]. Vaatimusten noudattamatta jättäminen voi johtaa oikeudellisiin seuraamuksiin, mainehaittoihin tai rajoitettuun pääsyyn rahoitusmarkkinoille, minkä vuoksi kestävyyden integrointi IT-strategioihin on monilla alueilla muuttunut vapaaehtoisesta toimenpiteestä sääntelyn edellyttämäksi käytännöksi.

Markkinadynamiikka suosii yhä enemmän kestäviä ratkaisuja. Asiakkaat ja kuluttajat ovat entistä ympäristötietoisempia ja odottavat

yrityksiltä uskottavia toimia kestävyyden edistämiseksi. McKinsey & Companyn globaalin analyysin mukaan ESG-tekijät voivat vaikuttaa positiivisesti asiakkaiden luottamukseen ja ostopäätöksiin [9]. Samansuuntaisesti PwC:n Global Consumer Insights Pulse Survey osoittaa, että suuri osa kuluttajista on valmis muuttamaan ostokäyttäytymistään ympäristö- ja vastuullisuussyistä, ja että yritysten kestävyyspyrkimykset vaikuttavat brändimielikuvaan ja asiakasuskollisuuteen [10]. Organisaatiot, jotka kykenevät osoittamaan energiatehokkaita IT-ratkaisuja ja vähähiilisiä digitaalisia palveluja, voivat siten saavuttaa kilpailuetua ja vahvistaa sidosryhmäsuhteitaan.

Lähteet

[1] The Shift Project. (2019). Lean ICT: Towards digital sobriety. Haettu: https://theshiftproject.org/en/publications/lean-ict

[2] International Energy Agency (IEA). (2017). Digitalisation and Energy. Haettu: https://www.iea.org/reports/digitalisation-and-energy

[3] International Telecommunication Union (ITU) & World Bank. (2024). Measuring the Emissions & Energy Footprint of the ICT Sector. Haettu: https://www.itu.int

[4] Freitag, C., Berners-Lee, M., Widdicks, K., Knowles, B., Blair, G., & Friday, A. (2021). The climate impact of ICT: A review of estimates, trends and regulations. arXiv. Haettu: https://arxiv.org/abs/2102.02622

[5] Natural Resources Defense Council (NRDC). (2014). Data Center Efficiency Assessment: Scaling Up Energy Efficiency Across the Data Center Industry. Haettu: https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IP.pdf

[6] U.S. Environmental Protection Agency (EPA) / ENERGY STAR. (2012). Understanding and Designing Energy-Efficiency Programs for Data Centers. Haettu: https://www.energystar.gov

[7] Mirantis. (2025). Top 13 Benefits of Virtualization for Enterprises. Haettu: https://www.mirantis.com/blog/top-13-benefits-of-virtualization-for-enterprises/

[8] EUR-Lex. (2022). Directive (EU) 2022/2464 – Corporate Sustainability Reporting Directive (CSRD). Haettu: https://eur-lex.europa.eu/eli/dir/2022/2464/oj/eng

[9] McKinsey & Company. (2020). How ESG creates value. Haettu: https://www.mckinsey.com/business-functions/strategy-and-corporate-finance/our-insights/how-esg-creates-value

[10] PwC. (2022). Global Consumer Insights Pulse Survey. Haettu: https://www.pwc.com/gx/en/issues/consumer-insights/global-consumer-insights-pulse-survey.html

2

Kestävä ohjelmistokehitys lyhyesti

VISIIRI.

Euroopan unionin osarahoittama

Kestävä ohjelmisto-kehitys lyhyesti

Ympäristökestävä ohjelmisto keskittyy minimoimaan ohjelmistojen ja niiden käyttämän laitteiston energia- ja resurssikulutuksen. Ympäristökestävä ohjelmistokehitys laajentaa perinteisen suorituskyvyn optimoinnin näkökulmaa tarkastelemalla ohjelmistojen ekologista vaikutusta kehityksestä käyttöönottoon ja poistoon saakka.

Ohjelmistot kuluttavat energiaa kaikissa niiden käyttövaiheissa. Tämän vaikutuksen minimoimiseksi ohjelmistokehittäjät voivat hyödyntää hiilitietoista ohjelmointia, eli rakentaa ratkaisuja, jotka ovat energian ja hiilen näkökulmasta tehokkaita (esim. vähentävät turhaa laskentaa ja datansiirtoa) [1]. Tämä voi tarkoittaa suorituskykyisempien algoritmien hyödyntämistä, tietorakenteiden optimointia ja tarpeettomien datasiirtojen vähentämistä. Tutkimuskirjallisuudessa korostetaan, että ohjelmiston arkkitehtuuri-, suunnittelu- ja toteutusratkaisut vaikuttavat suoraan sen energiankulutukseen ja resurssitehokkuuteen, ja huonot laaturatkaisut voivat lisätä ympäristövaikutuksia koko ohjelmiston elinkaaren ajan [2].

Energiankulutuksen lisäksi kestävä ohjelmisto pyrkii vähentämään fyysisten resurssien käyttöä. Tämä sisältää mm. datan tallennuksen optimoinnin pakkausta, jotta tarvittavien palvelinten ja kiintolevyjen määrä vähenisi. Se tarkoittaa myös kevyiden sovellusten rakentamista, mikä pidentää laitteiden käyttöikää ja vähentää elektroniikkajätettä (e-jäte) [3].

Kevyet sovellukset, jotka toimivat vanhemmillakin laitteilla, vähentävät laitteistopäivitysten tarvetta. Lisäksi kehittäjät voivat hyödyntää serverittömiä ratkaisuja ja kontteja, jotka mahdollistavat jaetun kapasiteetin tehokkaamman käytön, vähentäen tyhjillään olevien palvelinten määrää ja fyysistä laitteistojalanjälkeä [4].

Hiilitietoinen ohjelmistojen käyttöönotto liittyy ohjelmiston käyttöön sen operatiivisessa vaiheessa. Siihen kuuluu niin kutsuttu kysynnän ohjaus (demand shaping) ja kuorman siirtäminen ajassa/paikassa (demand shifting), eli laskentatehtävien ajoittaminen ja/tai sijoittaminen ajankohtiin ja alueille, jolloin sähköntuotannon hiili-intensiteetti on matalampi [5]. Esimerkiksi suuri datankäsittelytehtävä

voidaan ajoittaa ajankohtaan, jolloin uusiutuvan energian osuus verkossa on suurempi.

Tämä lähestymistapa hyödyntää hiili-intensiteetin käsitettä, joka mittaa hiilidioksidipäästöjen määrää tuotettua sähköyksikköä kohti. Sijoittamalla ohjelmistojen suorituspaikat alueille, joissa sähköntuotannon hiili-intensiteetti on matala, ja ajoittamalla tehtävät vastaavasti, organisaatiot voivat merkittävästi vähentää epäsuoria päästöjään [6].

Green Software Lifecycle (GSLC)

Green Software Life Cycle (GSLC) on viitekehys, joka integroi ympäristökestävyyden periaatteet kaikkiin ohjelmistokehityksen ja -toiminnan vaiheisiin. Toisin kuin perinteiset lähestymistavat, jotka keskittyvät pääasiassa toiminnallisuuteen, kustannuksiin ja markkinoilletuontiaikaan, GSLC huomioi nimenomaisesti ohjelmistojen ekologisen jalanjäljen aina hankinnasta poistoon saakka.
Vaikka ohjelmisto on aineeton, se kuluttaa merkittävästi laitteistoa ja energiaa koko elinkaarensa aikana. Siksi ohjelmistojen vihreämmäksi tekeminen edellyttää paitsi tehokasta koodia myös harkittuja päätöksiä suunnittelussa, kehityksessä, käyttöönotossa ja poistossa.

GSLC rakentuu vakiintuneiden ohjelmistokehitysmallien (SDLC) päälle lisäämällä kestävyyden ydinlaaduksi. Se laajentaa perinteisiä vaiheita (vaatimukset, suunnittelu, rakentaminen, testaus ja ylläpito) lisäämällä mukaan hankinta- ja poistovaiheet. Tämä laajennus pohjautuu kiertotalouden periaatteisiin ja pyrkii minimoimaan resurssien käytön ja jätteen määrän.

GSLC:n keskeinen käsite on kestävyysvelka, joka on analoginen teknisen velan kanssa: se kuvaa nykyisen kestävyyssuorituskyvyn ja ihannetilan välistä kuilua. Tämän velan tunnistaminen ja aktiivinen hallinta auttaa organisaatioita välttämään piileviä ympäristökustannuksia, jotka vaikeutuvat ja kallistuvat ajan myötä.

Käytännössä GSLC rohkaisee ohjelmistoalan ammattilaisia soveltamaan konkreettisia vihreitä käytäntöjä ja mittareita jokaisessa ohjelmiston elinkaaren vaiheessa. Esimerkiksi vaatimusmäärittelyssä määritellään selkeät energiankulutustavoitteet ja laitteistoyhteensopivuus, jotka pidentävät laitteiden käyttöikää ja vähentävät e-jätettä. Suunnittelussa ja kehityksessä pyritään kevyisiin arkkitehtuureihin, tehokkaisiin algoritmeihin ja uudelleenkäytettäviin

komponentteihin laskennan kuormituksen minimoimiseksi. Testauksen ja käytön aikana energiankulutusta seurataan ja optimoidaan jatkuvasti. Lopuksi poistovaiheessa varmistetaan ohjelmiston hallittu deaktivointi, jolloin data ja järjestelmäresurssit kierrätetään tai siirretään eteenpäin tarpeettoman jätteen välttämiseksi. [7]

Lähteet

[1] The Green Software Foundation. Principles of Green Software Engineering. https://greensoftware.foundation/

[2] Naumann, S., Dick, M., Kern, E., & Johann, T. (2011). The GREENSOFT Model: A reference model for green and sustainable software and its engineering. Haettu: https://doi-org.ezproxy.cc.lut.fi/10.1016/j.suscom.2011.06.004

[3] The Shift Project. (2019). *Lean ICT: Towards Digital Sobriety*. https://theshiftproject.org/en/

[4] McKinsey & Company. (2020). Cloud 2.0: Serverless architecture and the next wave of enterprise offerings. Haettu: https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/tech-forward/cloud-20-serverless-architecture-and-the-next-wave-of-enterprise-offerings

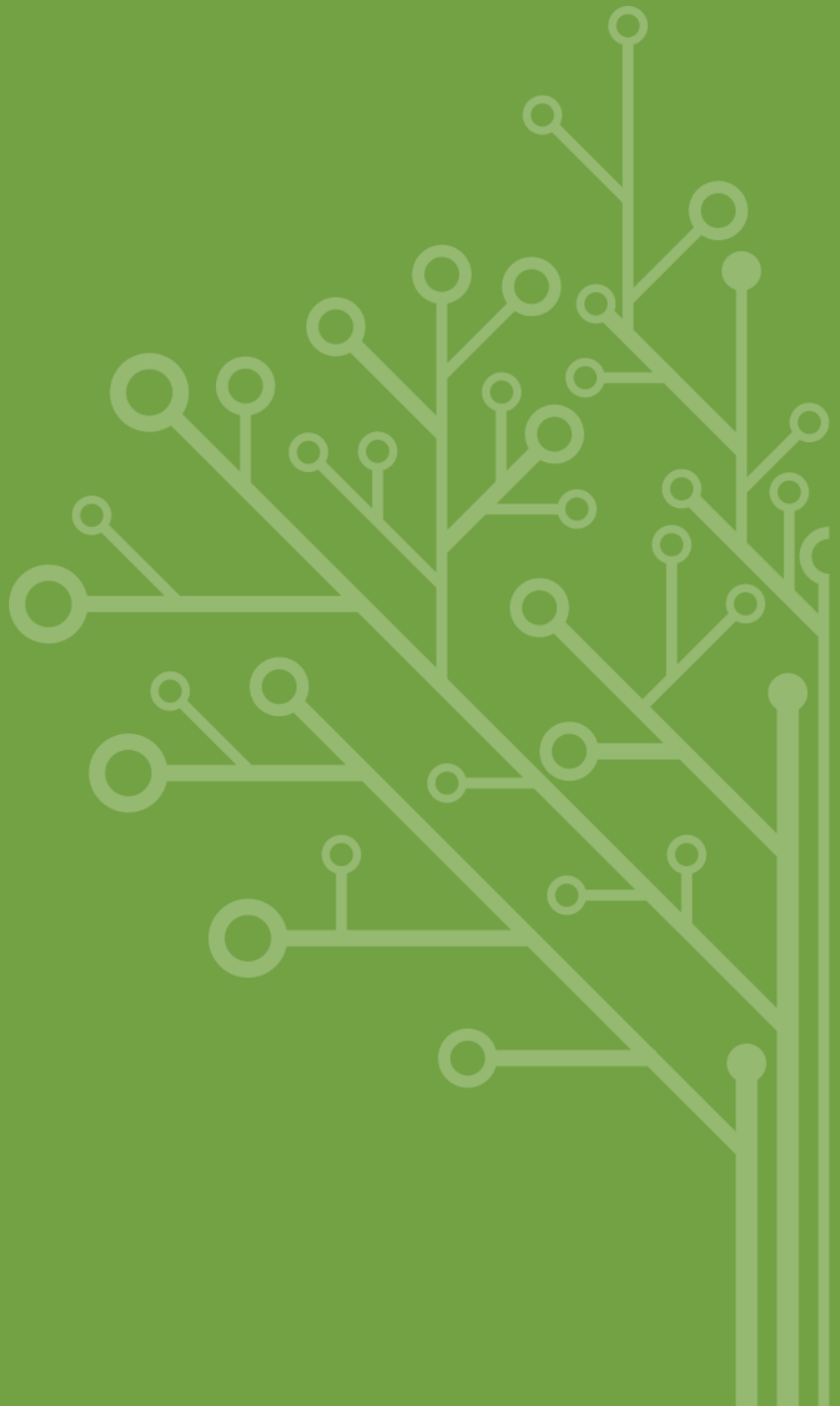
[5] Green Software Foundation. Carbon Awareness. Haettu: https://learn.greensoftware.foundation/carbon-awareness/

[6] Google. (2021). Using location to reduce our computing carbon footprint. Haettu: https://blog.google/company-news/outreach-and-initiatives/sustainability/carbon-aware-computing-location/

[7] L. Kivimäki *et al.*, "Building Up Green Software Life Cycle Model," *2024 10th International Conference on ICT for Sustainability (ICT4S)*, Stockholm, Sweden, 2024, pp. 20-28, Haettu: https://doi.org/10.1109/ICT4S64576.2024.00012

3

Data osana ohjelmistokehityksen elinkaarta



VISIIRI.

Euroopan unionin osarahoittama

Data osana ohjelmistokehityksen elinkaarta

Dataa voidaan ajatella ohjelmiston “polttoainena”, joka määrittää sekä palvelun toiminnallisuuden että merkittävän osan sen ympäristövaikutuksista. Jokainen datapiste aiheuttaa kuormaa vähintään neljässä vaiheessa: keruu → siirto → käsittely → tallennus (ja usein lisäksi uudelleenkäyttö, jakaminen sekä varmistukset).

Kun dataa kerätään “varmuuden vuoksi”, kustannukset ja ympäristövaikutukset kumuloituvat helposti, koska dataa replikoidaan, säilytetään pitkään ja käytetään yhä uusissa analytiikka- ja AI-käyttötapauksissa. Tämän vuoksi datan rooli on syytä käydä läpi ohjelmiston elinkaaren aikana samalla tarkkuudella kuin suorituskyky tai tietoturva.

Datan ympäristövaikutuksia pienennetään tehokkaimmin jo ennen varsinaista toteutusta määrittelemällä mitä dataa oikeasti tarvitaan (minimointi), millä tarkkuudella ja kuinka usein sitä tarvitaan, mikä on datan käyttötarkoitus ja mikä on sen säilytysaika. Minimointi ei ole pelkkä juridinen tai eettinen periaate, vaan myös tekninen keino vähentää kuormaa: vähemmän kerättyä dataa tarkoittaa vähemmän siirtoa, laskentaa ja tallennusta. Samalla kannattaa suunnitella datan “poistuminen” (retention & deletion) osaksi arkkitehtuuria eikä jälkikäteen lisättäväksi toiminnoksi.

Datan käsittelytavat näkyvät energiana erityisesti pilvi- ja konesaliympäristöissä. Globaalien datakeskusten energiankäyttö on ollut pitkään tehokkuusparannusten ansiosta yllättävän vakaata suhteessa työkuorman kasvuun, mutta kasvu- ja AI-ajurit lisäävät painetta, jolloin datavalintojen merkitys korostuu. [2] Datan siirto ei myöskään ole ilmaista. Internetin datansiirron sähkönkulutuksen intensiteetistä (kWh/GB) on koottu ja harmonisoitu arvioita, jotka osoittavat, että liikennemäärien kasvaessa myös optimoinneilla (pakkaus, välimuistit, reunalaskenta, tarpeettomien siirtojen välttäminen) on merkitystä. [1]

Datan elinkaari jatkuu tuotannossa, jossa lokit, telemetria, analytiikka ja AI-koulutus-/hienosäätödatat voivat kasvaa nopeasti, ellei niille ole

omistajuutta, laatukriteerejä ja elinkaarisääntöjä. Käytännöllinen periaate on “mittaa tarkoituksella”, eli kerää vain sellaista havaintodataa, jolla on selkeä päätöksentekokäyttö, ja tarkista säännöllisesti mittaristo ja säilytysajat. Datakäytännöt kannattaa dokumentoida ja standardoida, jotta data pysyy löydettävänä ja uudelleenkäytettävänä hallitusti (esim. FAIR-periaatteiden suuntaisesti) ilman tarpeetonta keruuta ja varastointia. [6]

Kestävyysajatteluun kuuluu myös datan “eläkkeelle jäämisen” eli poistopolut, anonymisointi/pseudonymisointi, arkistointi ja järjestelmien alasajo tulee suunnitella. Ilman eläkesuunnitelmaa vanha data jää elämään varastoihin ja varmistuksiin, mikä kasvattaa sekä kustannuksia että kuormaa.

Lähteet

[1] Aslan, J., Mayers, K., Koomey, J. G., & France, C. (2018). *Electricity Intensity of Internet Data Transmission: Untangling the Estimates.* Journal of Industrial Ecology, 22(4), 785–798. https://doi.org/10.1111/jiec.12630

[2] Masanet, E., Shehabi, A., Lei, N., Smith, S., & Koomey, J. (2020). *Recalibrating global data center energy-use estimates.* Science, 367(6481), 984–986. https://doi.org/10.1126/science.aba3758

[3] Freitag, C., Berners-Lee, M., Widdicks, K., Knowles, B., Blair, G. S., & Friday, A. (2021). *The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations.* Patterns, 2(9), 100340. https://doi.org/10.1016/j.patter.2021.100340

[4] Kern, E., Dick, M., Naumann, S., & Guldner, A. (2015). Environmental Impact Assessment Review, vol. 52, pages 53–61, DOI: 10.1016/j.eiar.2014.07.003

[5] Koot, M., & Wijnhoven, F. (2021). *Usage impact on data center electricity needs: A system dynamic forecast model.* Applied Energy, 291, 116798. https://doi.org/10.1016/j.apenergy.2021.116798

[6] Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., et al. (2016). *The FAIR Guiding Principles for scientific data management and stewardship.* Scientific Data, 3, 160018. https://doi.org/10.1038/sdata.2016.18

4

Mitä tulee huomioida ohjelmiston elinkaaren eri vaiheissa?

VISIIRI.

Euroopan unionin
osarahoittama

Vaihe 1: Hankinta

Kun organisaatiot hankkivat ohjelmistoja, kestävyys tulisi nähdä keskeisenä kriteerinä eikä jälkikäteen lisättävänä huomiona. Tämä tarkoittaa ohjelmistojen energiatehokkuuden tarkastelua, sillä algoritmien suunnittelu, siirretyn datan määrä ja käyttöliittymien monimutkaisuus vaikuttavat siihen, kuinka paljon energiaa kuluu sekä päivittäisessä käytössä että huippukuormituksen aikana. Yhtä tärkeää on varmistaa, että ohjelmisto toimii hyvin olemassa olevalla laitteistolla pakottamatta tiheisiin laitepäivityksiin. Tämä pidentää laitteiden käyttöikää ja vähentää elektroniikkajätettä.

Toimittajien olisi ihanteellisesti tarjottava selkeää tietoa päästöistä tai energiankulutuksesta sekä työkaluja, joiden avulla organisaatiot voivat mitata suorituskykyä itse. Kestävän hankinnan käytännöt sitovat nämä näkökulmat yhteen, varmistaen että ympäristövaikutukset arvioidaan kustannusten, turvallisuuden ja toiminnallisuuden rinnalla.

Organisaatiot voivat integroida kestävyysvaatimuksia hankintaprosesseihinsa ja määritellä selkeitä suoritusindikaattoreita, kuten hyväksyttävät energiankulutustasot tai datansiirron rajat. Sellaisen ohjelmiston valitseminen, joka on modulaarinen, mukautuva eikä pakota laitteiston vaihtoon, voi vähentää ympäristövaikutuksia koko elinkaaren ajan.

Ostajilla on myös valtaa vaikuttaa markkinoihin palkitsemalla toimittajia, jotka osoittavat ympäristövastuullisia käytäntöjä, ja vaatimalla aiempaa suurempaa läpinäkyvyyttä. Kun ohjelmiston kokonaisjalanjälki huomioidaan hankinnasta päivityksiin, ylläpitoon ja lopulliseen poistoon, painopiste siirtyy lyhyen aikavälin säästöistä pitkäjänteiseen kestävyyteen. Tietoisuuden lisääminen kehittäjien, toimittajien ja loppukäyttäjien keskuudessa vahvistaa tätä muutosta, auttaen varmistamaan, että ohjelmistohankinnasta tulee aktiivinen tapa vähentää digitaalisten teknologioiden ympäristökuormaa.

1. Kestävyys vaatimuksena
    - Sisällytä selkeät kestävyys- ja energiatehokkuuskriteerit tarjouspyyntöihin (RFP) ja arviointimatriiseihin.
    - Pyydä kvantitatiivista dataa odotetusta energiankulutuksesta, hosting-infrastruktuurista ja hiili-intensiteetistä.

2. Jaettu vastuu

- Määrittele, mitkä osapuolet (toimittaja, pilvipalveluntarjoaja, asiakas) vastaavat päästöjen seurannasta ja vähentämisestä.
- Sisällytä ympäristömittarit ja vastuut palvelutasosopimuksiin (SLA).

3. Läpinäkyvyys ja raportointi
    - Edellytä jatkuvaa mittaamista ja säännöllistä raportointia energiankulutuksesta ja ympäristöindikaattoreista.
    - Hyödynnä telemetriadataa tai standardoituja mittareita (esim. energia per transaktio, hiili per käyttäjäistunto).

Vaihe 2: Vaatimukset

Jo varhaisista suunnitteluvaiheista lähtien vaatimusten tulisi pyrkiä minimoimaan tallennettava, siirrettävä tai käsiteltävä data, sillä suuret datamäärät lisäävät energiankulutusta, verkkoliikennettä ja laitteiden tallennuskapasiteetin kuormitusta. Tämä sisältää staattisen datan ja analytiikkadatan rajoittamisen, datamallien yksinkertaistamisen, tehokkaiden protokollien valinnan, tarpeettomien käyttäjäsyötteiden välttämisen sekä sen varmistamisen, että vain olennaiset ominaisuudet ja kirjastoriippuvuudet sisällytetään ohjelmistoon.

Vaatimusten tulisi myös varmistaa, että ohjelmisto toimii tehokkaasti vaihtelevissa laitteisto- ja verkko-olosuhteissa. Koska sama ohjelmisto voi kuluttaa huomattavasti eri määriä energiaa eri alustoilla ja erilaisissa verkkoympäristöissä, vaatimuksissa on määriteltävä sopeutumiskyky. Esimerkiksi ohjelmiston tulisi pystyä skaalautumaan alaspäin tai poistamaan tietyt ominaisuudet käytöstä silloin, kun kaistanleveys on rajallinen tai kun käytetään mobiilidataa kiinteän yhteyden sijaan. Oletusasetusten tulisi suosia energiatehokkaita vaihtoehtoja, ja käyttäjiä tulisi informoida tilanteista, joissa tietyt ominaisuudet kuluttavat paljon dataa, sekä antaa heille mahdollisuus hallita näitä ominaisuuksia.

Ohjelmisto ei myöskään saa aiheuttaa laitteiston ennenaikaista vanhenemista. Hankinta- tai kehitysspesifikaatioihin tulisi sisällyttää vaatimus yhteensopivuudesta vanhempien tai vähemmän tehokkaiden laitteiden kanssa aina kun mahdollista, jotta laitteiden käyttöikä pitenee ja laitteiden valmistukseen ja hävittämiseen liittyvät päästöt pienenevät.

Vaatimusten tulisi myös käsitellä koodin tehokkuutta: sopivien ohjelmointikielten valintaa, algoritmien ja tietorakenteiden optimointia, ylimääräisten tai paisuneiden kirjastojen käytön vähentämistä sekä harvoin käytettyjen ominaisuuksien poistamista. Optimoinnin ja refaktoroinnin tulee olla osa spesifikaatiota, sekä ajonaikaisen energiankulutuksen pienentämiseksi että ohjelmistojätteen minimoimiseksi.

1. Teknologiastrategia ensin

- Arvioi organisaation nykyiset kyvykkyydet ja tunnista alueet, joilla tarvitaan uutta osaamista.
- Valitse teknologiat, jotka tasapainottavat toimittajariippumattomuuden, tiimin osaamisen ja pitkän aikavälin ylläpidettävyyden.
- Dokumentoi perustelut kaikille valinnoille: tulevien tiimien tulee ymmärtää miksi, ei vain mitä.
- Sisällytä kestävyyskriteerit teknologiapäätöksiin (esim. energiatehokkuus, avoimen lähdekoodin kypsyys, elinkaarivaikutukset).

2. Arkkitehtuuri skaalautuvuudelle ja siirrettävyydelle

- Määrittele jo varhain, missä ratkaisu tulee toimimaan: julkisessa pilvessä, omassa konesalissa vai hybridi-/monipilviympäristössä.
- Käytä konttipohjaisia alustoja (esim. Kubernetes) yhtenäisten käyttöönottojen mahdollistamiseksi eri ympäristöissä.
- Suunnittele joustavuutta varten, ei rajatonta skaalautumista ja varmista linjaus realististen liiketoiminnan kasvuskenaarioiden kanssa.
- Varmista arkkitehtuuripäätösten palautettavuus, jotta pitkän aikavälin tekninen velka vältetään.

3. Datastrategia minimalismin periaatteella

- Keskitetty datan tallennus ja vain tarpeellisen datan integrointi, tarpeettoman duplikaation välttäminen.
- Toteuta kysyntään perustuva (on-demand) datan käyttö vähentämään järjestelmäkuormaa ja energiankulutusta.
- Määrittele datan omistus, hallinta ja elinkaari ensimmäisestä päivästä alkaen.
- Minimoi säilytysaika ja anonymisoi data mahdollisuuksien mukaan: kestävyys tarkoittaa myös digitaalista vastuullisuutta.

4. Tekoälyn vastuullinen käyttö

- Hyödynnä tekoälyä suunnittelun, kehityksen ja testauksen vauhdittamiseen, ei ylituotantoon tai tarpeettoman koodin lisäämiseen.

- Noudata fail-fast-periaatetta ja iteratiivista lähestymistapaa, jotta ideat voidaan validoida varhain.
- Pidä tekoälyn tuottama sisältö jäljitettävänä ja dokumentoi keskeiset kehote- ja logiikkavalinnat.
- Säilytä human-in-the-loop varmistaaksesi laadun, läpinäkyvyyden ja eettisen käytön.

5. Hallintamalli ja jatkuva oppiminen

- Perusta poikkitoiminnallinen ohjausmalli: tuote-, arkkitehtuuri- ja kestävyysvastuulliset jakavat vaatimusten omistajuuden.
- Kerää jatkuvaa palautetta piloteista ja käyttäjiltä vaatimusten tarkentamiseksi ja hukan välttämiseksi.
- Edistä oppimisen ja iteroinnin kulttuuria: jokaisen kehityssyklin tulisi parantaa sekä koodia että yhteistyötä.

Vaihe 3: Suunnittelu ja toteuttaminen

Ohjelmiston suunnittelu vaikuttaa merkittävästi digitaalisten teknologioiden ympäristöjalanjälkeen, muovaten päästöjä ohjelmiston suorituksen aikana ja epäsuorasti myös laitteiston, verkkojen ja käyttäjien toiminnan kautta. Vaikka ohjelmisto ei tuota fyysistä elektroniikkajätettä eikä kuluta raaka-aineita samalla tavalla kuin laitteisto, sen suunnittelu voi pakottaa käyttäjät päivittämään laitteitaan tai lisäämään energiankulutusta. Tämä kaikki kiihdyttää valmistukseen ja hävittämiseen liittyviä piileviä päästöjä. Tämän vuoksi kestävän ohjelmistosuunnittelun on huomioitava sekä suorat että epäsuorat ympäristövaikutukset.

Keskeinen periaate ympäristötietoisessa suunnittelussa on datan minimointi. Jo ohjelmistoprojektin alkuvaiheessa suunnittelijoiden tulisi rajoittaa tallennettavan, siirrettävän ja käsiteltävän datan määrää. Suuret datamallit, tiheät siirrot tai raskas staattinen ja analytiikkadata lisäävät energiankulutusta ja verkon kuormitusta; kun toiminnot on kerran suunniteltu vaatimaan suuria datamääriä, mahdollisuudet merkittäviin optimointeihin vähenevät. Suunnittelun tulisi ennakoida erilaiset verkko-olosuhteet mukauttamalla ohjelmiston toimintaa tilanteen mukaan esimerkiksi vähentämällä tai poistamalla käytöstä runsaasti dataa kuluttavia ominaisuuksia mobiiliverkoissa kiinteiden yhteyksien sijaan. Oletusasetusten tulisi suosia energiatehokkaita tiloja. Käyttäjille tulisi kertoa dataraskaista toiminnoista ja antaa mahdollisuus vaikuttaa niihin.

Toinen olennainen suunnitteluperiaate on yhteensopivuus olemassa olevan laitteiston kanssa. Ohjelmiston ei tulisi aiheuttaa tarpeettomia laitepäivityksiä; sen resurssivaatimusten tulisi olla kohtuullisia myös vanhemmilla tai vähemmän tehokkailla laitteilla. Näin ohjelmisto auttaa pidentämään laitteiden käyttöikää, vähentäen paitsi käytönaikaista energiankulutusta myös laitteiden valmistukseen ja hävittämiseen liittyviä piileviä päästöjä.

Koodin arkkitehtuurin, algoritmivalintojen ja riippuvuuksien tehokkuus on niin ikään keskeistä. Suunnittelussa on valittava energiatehokkaita algoritmeja ja tietorakenteita, karsittava ylimääräinen koodi, poistettava harvoin käytetyt kirjastot tai ominaisuudet sekä huomioitava suorituskonteksti. Mahdollisuuksien mukaan komponenttien tulisi olla modulaarisia ja kevyitä. Suunnitteluratkaisuissa tulisi suosia kehyksiä, kieliä ja arkkitehtuureja, joiden tiedetään kuluttavan vähemmän energiaa sekä kehityksen että suorituksen aikana.

Lopuksi käyttöliittymän ja käyttäjäkokemuksen suunnittelu vaikuttaa siihen, kuinka paljon energiaa ja dataa kuluu. Visuaaliset elementit (esim. animaatiot, korkearesoluutioiset grafiikat ja tarpeettoman monimutkaiset asettelut) lisäävät tiedonsiirtoa ja prosessointia ja voivat johtaa väärinkäsityksiin, jotka pakottavat käyttäjät toistamaan toimintoja, mikä kasvattaa energiankulutusta. Yksinkertaiset asettelut, selkeät toiminnot ja "visuaalisen melun" minimointi eivät ole pelkästään esteettisiä kysymyksiä, vaan tukevat kestävyyttä. Selkeä palaute, tarpeettomien vuorovaikutusten minimointi ja energiatehokkaat oletusasetukset auttavat kaikki vähentämään hukkaa.

Suunnitteluperiaate	Toimenpiteet onnistumiseen	Odotettu vaikutus
Datan minimointi	Suunnittele API:t ja datamallit, jotka rajoittavat tallennettavan ja siirrettävän datan määrää. Hyödynnä välimuistia, pakkausta ja tapahtumapohjaisia päivityksiä.	Pienempi datansiirron energiankulutus ja palvelinkuorma.
Laitteistoyhteensopivuus	Varmista suorituskyky vanhemmilla tai vähäisillä laitteistoresursseilla; testaa hallittu toiminnan heikkeneminen.	Pidentää laitteiden käyttöikää → vähentää valmistukseen liittyviä päästöjä.
Tehokas arkkitehtuuri ja koodi	Käytä modulaarisia ja kevyitä kehyksiä; poista käyttämättömät kirjastot; ajoita laskentatehtäviä matalan hiili-intensiteetin ajankohtiin.	20–40 % pienempi infra- ja ohjelmistoenergian-kulutus.
Käyttökokemus ja käyttöliittymän tehokkuus	Minimoi animaatiot, ota käyttöön eko-tilat/tummat teemat ja ohjaa käyttäjiä energiatietoiseen toimintaan.	Pienempi käyttöliittymän energiankulutus ja parempi käytettävyys.
Mittaaminen ja iterointi	Hyödynnä CO_2.js-, Impact Framework- ja pilven hiilipäästödashboard-työkaluja osana kehityssyklejä.	Jatkuva näkyvyys tiimeille ja PO:ille → jatkuva parantuminen.

Vaihe 4: Testaaminen

Ohjelmistotestauksella on ympäristövaikutuksia, jotka ulottuvat sen välittömän tehtävän, eli oikeellisuuden ja luotettavuuden varmistamisen ulkopuolelle. Testaus kuluttaa laskentaresursseja, tapahtuipa se automatisoitujen testipakettien toistuvana ajamisena palvelimilla tai manuaalisina testausvaiheina kehittäjien työasemilla. Jokainen testin suoritus vaatii energiaa, ja kun testausympäristöjä skaalataan useille laitteille tai jatkuvan integraation putkiin, kokonaiskuorma kasvaa merkittävästi. Tämä resurssien käyttö heijastaa ohjelmiston ja laitteiston välistä laajempaa suhdetta: liialliset tai tarpeettomat testiajot eivät ainoastaan lisää suoraa energiankulutusta, vaan voivat myös nopeuttaa infrastruktuurin kulumista, mikä puolestaan lisää laitteiden vaihtotarvetta ja siihen liittyviä päästöjä.

Samalla testaus tarjoaa tärkeän mahdollisuuden tunnistaa ja korjata tehottomuuksia ennen kuin ne juurtuvat tuotantojärjestelmiin. Tehoton koodi, tarpeeton datankäsittely ja huonosti optimoidut algoritmit eivät ole ainoastaan suorituskykyongelmia, vaan myös ympäristökuormia, sillä ne lisäävät ohjelmiston energiankulutusta käytön aikana. Tehokkaat testausmenettelyt voivat paljastaa tällaisia ongelmia ja mahdollistaa arkkitehtuurien keventämisen, tallennus- ja siirtotarpeiden vähentämisen sekä tarpeettomien prosessien poistamisen. Tässä mielessä testauksen ympäristökustannuksia voidaan tasapainottaa pitkän aikavälin resurssisäästöillä, jotka saavutetaan parantuneen ohjelmistotehokkuuden kautta.

Myös testausympäristöjen kokoonpano vaikuttaa kestävyystuloksiin. Ohjelmistojen suorituskyky ja energiankulutus vaihtelevat laitteiden ja alustojen mukaan, joten testaaminen monipuolisilla laitteistoprofiileilla varmistaa, että energiaintensiivinen käyttäytyminen tunnistetaan eri olosuhteissa. Tämä monimuotoisuus auttaa estämään sellaisten ohjelmistojen julkaisun, jotka ovat tehokkaita yhdellä järjestelmällä mutta hukkaavat resursseja toisella, ja siten tukee laitteiden pidempiä käyttöikiä vähentämällä tarvetta tarpeettomiin päivityksiin. Laajat testipaketit tarjoavat kattavampaa näkyvyyttä ja voivat paljastaa syvempiä tehottomuuksia, mutta ne kuluttavat myös enemmän energiaa. Kestävät käytännöt edellyttävät siksi tarkkaa tasapainottamista. Testauksen tulee olla riittävän kokonaisvaltaista laadun takaamiseksi ja energiankulutusta lisäävien suunnitteluvirheiden havaitsemiseksi, mutta ei niin ylitsepursuavaa, että itse testaus tuottaa vältettävissä olevia päästöjä.

Vaihe 5: Käyttö ja ylläpito

Ohjelmistojen ylläpito ja operointi aiheuttavat jatkuvia ympäristövaikutuksia, jotka kumpuavat samoista elinkaaren dynamiikoista, jotka vaikuttavat laajemmin sekä laitteiston että ohjelmiston hiilijalanjälkeen. Ajan myötä, kun ohjelmistoja ylläpidetään (korjauspaketeilla, päivityksillä, bugikorjauksilla ja uusien ominaisuuksien käyttöönotolla), ne alkavat usein vaatia pohjalta toimivalta laitteistolta yhä enemmän prosessorisyklien, muistin, tallennustilan ja datansiirron osalta. Nämä vaatimukset kasvattavat käytönaikaista energiankulutusta, mikä lisää päästöjä huomattavasti, kun vaikutus kertautuu suurille käyttäjä- tai laitemäärille.

Myös operatiiviset käytännöt vaikuttavat resurssien kulutukseen: infrastruktuuri, joka mahdollistaa ohjelmistojen ajamisen (palvelimet datakeskuksissa, verkkoreitit, sisällönjakelupalvelut ja käyttäjien päätelaitteet), on pidettävä käynnissä, jäähdytettynä, virralla ja ylläpidettynä. Kun ohjelmisto ei ole optimoitu tai kun päivitykset tuovat mukanaan tehottomuutta, nämä operatiiviset järjestelmät saattavat tarvita useammin skaalautumista tai laitteistopäivityksiä, mikä edelleen kasvattaa valmistuksen ja myöhemmän hävittämisen kautta syntyviä piileviä päästöjä. Laitteiden käyttöikä onkin tiiviisti sidoksissa ohjelmiston vaatimuksiin: jos ohjelmisto kehittyy tavalla, joka tekee vanhemmasta laitteistosta riittämätöntä tai käyttökelvotonta, paine päivittää laitteita kasvaa ja samalla kasvaa ympäristökuorma.

Ylläpidon ja operoinnin toinen ulottuvuus liittyy datan tallennukseen ja siirtoon. Kun ohjelmistojärjestelmät kasvavat, myös säilytetyn, käsitellyn ja siirretyn datan määrä lisääntyy: lokit, analytiikka, varmuuskopiot ja käyttäjien tuottama sisältö kertyvät usein ilman kriittistä tarkastelua. Jokainen tallennettu tai siirretty datayksikkö kuluttaa energiaa tallennuksessa ja lisäksi kuormittaa verkko- ja palvelininfrastruktuuria sekä jäähdytysjärjestelmiä. Datan tarpeettomuus, tehottomat datamallit tai tarpeeton datan replikointi operatiivisissa järjestelmissä lisäävät energiantarvetta ja kasvattaa siten hiilijalanjälkeä.

Operointiin kuuluu myös käyttäjätoiminta, joka korostaa ylläpidon aikana tehtyjen ohjelmistopäätösten vaikutuksia. Jos esimerkiksi päivitykset ottavat oletuksena käyttöön resurssi-intensiivisiä ominaisuuksia tai uudet ominaisuudet ovat automaattisesti aktivoituina, monet käyttäjät kokevat lisääntynyttä energiankulutusta ilman vastaavaa hyötyä. Operatiiviset päätökset kuten se, tukeeko ohjelmisto taaksepäin yhteensopivuutta, annetaanko käyttäjille valinnanvaraa resurssien käyttöasteesta tai optimoidaanko toiminta

erilaisiin verkko-olosuhteisiin vaikuttavat ratkaisevasti ohjelmiston kestävyyteen sen käytön aikana.

Yleisesti ihmiset haluavat tehdä oikein ja vähentää resurssien kulutusta, mutta tämä vaatii aikaa ja keskittymistä. Monesti se tarkoittaa myös toimiluvan kysymistä: jos energiankulutuksen vähentäminen ei itsessään riitä perusteeksi, taloudelliset kannustimet voivat olla ratkaisevia. Yksi tehokas lähestymistapa on FinOps, erityisesti julkisissa pilvissä, joissa taloudellinen näkökulma toimii vipuna. Pienempi laskentayksikkö, joka suorittaa saman tehtävän, tarkoittaa sekä pienempää hintaa että pienempää hiilijalanjälkeä. Resurssikulutuksen tekemiseksi näkyväksi merkitään ensin resurssit ja mitataan niiden käyttöä ajan kuluessa. Kun samaan prosessiin liitetään CO_2e-data (hiilidioksidiekvivalentti), syntyy käytäntö, jota kutsutaan GreenOpsiksi. Edistystä kannattaa tehdä näkyväksi lisäämällä mittareita yhteisiin näkymiin ja tarvittaessa ottamalla käyttöön pelillisiä elementtejä vähennystyössä.

Kaiken tämän keskellä kestävä ylläpito lähtee liikkeelle vähentämisestä, ei mittaamisesta. Optimointi ei ole odottamista, vaan sellaisten asioiden poistamista, jotka eivät enää palvele tarkoitustaan: tarpeeton data, käyttämätön laskentakapasiteetti, liialliset uudelleensuoritukset ja paisuneet riippuvuudet. Tehokkuus kasvaa päätöksistä, jotka tehdään lähellä konkreettista työtä missä tiimit näkevät ja kokevat järjestelmän toiminnan. Mittaaminen on silti tärkeää, mutta heijastuksena: tapana oppia siitä, mitä on jo yksinkertaistettu. Kestävän käytön ja ylläpidon polku on iteratiivinen karsinta: toimi, havainnoi, paranna. Kun ylläpito suosii selkeyttä kasaamisen sijaan, sekä kustannukset että hiilijalanjälki pienenevät luonnollisina, kurinalaisen ohjelmistotekniikan sivutuotteina.

Vaihe 6: Poistaminen

Ohjelmistojen poistaminen käytöstä sisältää merkittäviä ympäristövaikutuksia, kun sitä tarkastellaan energiankulutuksen, datan säilytyksen ja järjestelmävaatimusten näkökulmasta. Toisin kuin fyysisiä laitteita, ohjelmistoja ei voida kierrättää perinteisessä mielessä, mutta niiden käytöstä poistaminen tai hylkääminen synnyttää epäsuoria vaikutuksia sekä infrastruktuuriin että käyttäjien toimintaan. Kun ohjelmistoa ei enää tueta, käyttäjät voidaan pakottaa siirtymään uudempien sovellusten tai käyttöjärjestelmien pariin. Tämä prosessi vaatii usein enemmän laskentaresursseja ja monissa tapauksissa myös uutta laitteistoa. Näin ohjelmiston poistaminen voi epäsuorasti lyhentää laitteiden käyttöikää ja vauhdittaa laitteiston päätymistä elinkaarensa loppuvaiheeseen.

Ohjelmistojen poistamisen toinen ulottuvuus liittyy digitaalisten jäänteiden, kuten tallennetun datan, tarpeettomaksi jääneiden kirjastojen ja käyttämättömien koodikantojen jatkuvuuteen. Jos näitä ei aktiivisesti poisteta käyttöönottovaiheessa tai alasajossa, ne jatkavat tallennustilan kuluttamista ja vaativat energiaa jatkuvien ylläpito- ja varmuuskopiointiprosessien kautta. Tämä osoittaa, että poistaminen ei ole pelkkä ohjelmiston käytön lopettaminen, vaan se edellyttää tietoista datan poistamista, riippuvuuksien purkamista ja tarpeettoman digitaalisen kuorman vähentämistä.

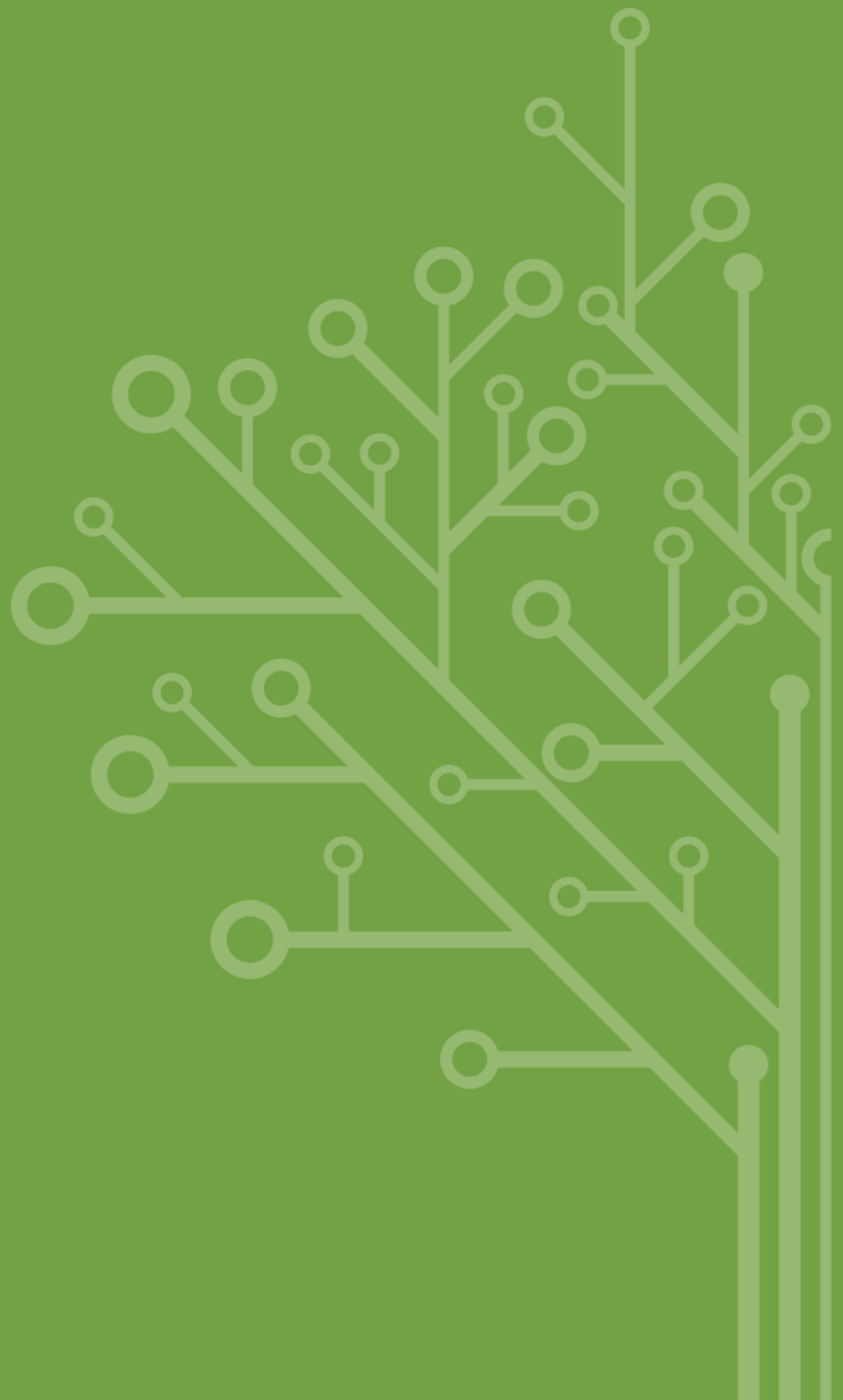
Ohjelmiston poistamiseen kytkeytyy myös yhteensopivuuden kysymys. Kun sovelluksia tai järjestelmiä vedetään pois käytöstä ilman taaksepäin yhteensopivuutta tai kestäviä vaihtoehtoja, käyttäjät ja organisaatiot voivat joutua äkillisiin siirtymiin, jotka vaativat laajamittaisia datasiirtoja ja järjestelmäkonfiguraatioiden uudelleenrakennusta. Nämä prosessit voivat olla laskennallisesti raskaita ja energiaintensiivisiä, mikä kasvattaa poistovaiheeseen liittyviä epäsuoria päästöjä.

Oleellista on myös huomioida miten olemassa oleva data siirretään uuteen järjestelmään. Huonosti suunniteltu tai tarpeettoman laaja datamigraatio voi lisätä merkittävästi energiankulutusta ja laskennallista kuormaa, kun taas harkittu datan karsinta, siirtotarpeen arviointi ja tehokkaiden siirtomenetelmien käyttö voivat vähentää siirtymävaiheen ympäristövaikutuksia.

Ohjelmiston käytöstä poistaminen korostaa elinkaaren aiemmissa vaiheissa tehtyjen suunnittelu- ja ylläpitovalintojen kumulatiivisia vaikutuksia. Ohjelmat, jotka edellyttävät jatkuvaa korvaamista tai aiheuttavat tarpeetonta vanhenemista, synnyttävät tiheämpiä poistokiertoja, kun taas kevyet, modulaariset ja tehokkaat sovellukset voivat pidentää käyttöikää.

5

Kestävän kulttuurin rakentaminen ohjelmistotiimiin



VISIIRI.

Euroopan unionin osarahoittama

Kestävän kulttuurin rakentaminen ohjelmistotiimiin

Organisaatiokulttuurin rakentaminen alkaa ylimmän johdon aidosta sitoutumisesta ja selkeästi määritellyistä arvoista, jotka työntekijät ymmärtävät ja kokevat merkityksellisiksi omassa työssään. Kun yhteiset arvot eivät jää vain kirjauksiksi, vaan näkyvät johdon päätöksissä ja päivittäisessä toiminnassa, luodaan vahva perusta kestävän kulttuurin kehittämiselle myös ohjelmistotiimeissä.

Kestävän kulttuurin rakentaminen ohjelmistotiimiin edellyttää huomiota sekä ympäristövastuuseen että kehitystyötä tekevien henkiseen hyvinvointiin. Ympäristön näkökulmasta kestävyys vahvistuu, kun tiimit tuovat tietoisuuden energiankulutuksesta ja laitteiden elinkaaresta osaksi arjen käytäntöjä. Päätökset koodin tehokkuudesta, datanhallinnasta ja järjestelmäarkkitehtuurista vaikuttavat paitsi suorituskykyyn myös ohjelmistojärjestelmien pitkän aikavälin hiilijalanjälkeen. Kun nämä näkökohdat sisällytetään luontevasti tiimin keskusteluihin, ekologinen kestävyys muuttuu luonnolliseksi osaksi ohjelmistokehittäjien ammatti-identiteettiä eikä jää irralliseksi lisähuomioksi.

Henkinen kestävyys on yhtä keskeinen osa kokonaisuutta. Ohjelmistokehitys on usein nopeasyklistä ja vaativaa, ja muuttuvat vaatimukset sekä suuri tehtäväkompleksisuus voivat kuormittaa yksilöitä. Tutkimukset osoittavat, että roolien selkeys, psykologinen turvallisuus ja autonomia ennustavat vahvasti positiivisia lopputuloksia ohjelmistokehittäjillä, kuten työtyytyväisyyttä ja hyvää suorituskykyä [2], [4]. Kulttuuri ja organisaatio joka tukee avoimuutta, luottamusta ja yhteistä ongelmanratkaisua, auttaa ehkäisemään uupumusta. Tämä on linjassa katsauksien kanssa, joiden mukaan insinöörien hyvinvointi rakentuu monista tekijöistä, kuten organisatorisesta tuesta aina tiimien sosiaalisiin suhteisiin [3].

On tärkeää huomata, että henkinen ja ympäristöllinen kestävyys eivät ole toisistaan erillisiä. Tiimillä, joka kokee olonsa turvalliseksi ja tuetuksi, on parempi edellytys tarttua luovasti ympäristöhaasteisiin ja rakentamaan pitkäjänteisiä kestäviä käytäntöjä [5]. Vastaavasti työskentely ympäristötavoitteiden eteen voi vahvistaa tiimin yhteistä tarkoitusta, mikä lisää motivaatiota ja työtyytyväisyyttä. Ketterän

ohjelmistokehityksen tutkimus korostaa myös sosiaalisen kestävyyden merkitystä, sillä teknisten tavoitteiden yhdistäminen ihmisten tarpeiden huomioimiseen tuottaa kestävämpiä ja tehokkaampia tiimejä [1].

Kestävän kulttuurin rakentaminen ohjelmistotiimiin on kokonaisvaltainen prosessi: se yhdistää ekologisen vastuun ja psykologiset edellytykset pitkäaikaiselle, hyvinvoivalle yhteistyölle. Näitä molempia vahvistamalla organisaatiot voivat varmistaa, että ohjelmistot ovat tehokkaita ja kestäviä ja että niiden kehittäjät pysyvät motivoituneina, luovina ja hyvinvoivina.

Lähteet

[1] Ahmad, M. O., et al. (2025). Fostering social sustainability in large-scale agile projects. Journal of Software: Practice & Experience. https://www.tandfonline.com/doi/full/10.1080/12460125.2025.2464749

[2] Buvik, M. P., & Tkalich, A. (2021). Psychological safety in agile software development teams: Work design antecedents and performance consequences. arXiv preprint arXiv:2109.15034. https://arxiv.org/abs/2109.15034

[3] Godliauskas, P., & Šmite, D. (2024). The well-being of software engineers: A systematic literature review and a theory. Empirical Software Engineering, 30(35).

https://link.springer.com/article/10.1007/s10664-024-10543-8
[4] Lenberg, P., & Feldt, R. (2018). Psychological safety and norm clarity in software engineering teams. arXiv preprint arXiv:1802.01378. https://arxiv.org/abs/1802.01378

[5] Montes, C. M., Penzenstadler, B., & Feldt, R. (2025). The factors influencing well-being in software engineers: A cross-country mixed-method study. arXiv preprint arXiv:2504.01787. https://arxiv.org/abs/2504.01787

Tarvitsetko apua?

Tämä käsikirja on tehty yhteistyössä seuraavien asiantuntijoiden kanssa, jotka ovat valmiita keskustelemaan aiheesta lisää tarpeen mukaan.

Advania, Tuomas Pasanen, www.advania.fi
Kanto Company, Tuomo Niemelä, www.kantocompany.com
Trail Openers, Ville Nordberg, www.trailopeners.com
LUT-Yliopisto, Ohjelmistotuotannon laitos, www.lut.fi

advania

KANTO COMPANY

Trail Openers

VISIIRI – Vihreän siirtymän ICT-ekosysteemi

Green ICT -hanke VISIIRI luo kokonaiskuvan Suomen ICT-alan vaikutuksista ilmastoon ja ympäristöön. Hanke tukee ICT-alan vihreää siirtymää yhdistämällä alan toimijat valtakunnalliseen ekosysteemiin, joka mahdollistaa parhaiden käytäntöjen jakamisen ja suomalaisen teollisuuden ja akateemisten toimijoiden kohtaamisen.

Hanke kehittää menetelmiä ICT-alan ympäristövaikutusten mittaamiseen ja tuottaa ympäristötietoisuutta lisääviä koulutusmateriaaleja yritysten käyttöön. Ympäristötietoisuuden lisääminen pienentää ICT-alan hiilijalanjälkeä, ja samaan aikaan kädenjälki suurenee. Samalla vihreä bisnes luo Suomeen edelläkävijyyttä, joka avaa mahdollisuuksia kansainvälisillä markkinoilla.

Hankeaika

01.04.2024–31.03.2026

Lisätietoja

tieke.fi/green-ict-visiiri