Vihreä ohjelmointi

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
2024
Tomas Saarinen

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

Tietotekniikan laitos, Teknillinen tiedekunta
Turun yliopisto

Oppiaine: Tieto- ja viestintätekniikka
Tutkinto-ohjelma: Tieto- ja viestintätekniikka
Tekijä: Tomas Saarinen
Otsikko: Vihreä ohjelmointi
Sivumäärä: 20 sivua
Päivämäärä: Lokakuu 2024

Tässä tutkielmassa selvitetään ICT-alan energiankulutusta ja erityisesti ohjelmakoodin vaikutusta siihen sekä sitä, miten ohjelmoija voi itse vaikuttaa työnsä energiankulutukseen. ICT-alaa, sen energiankulutusta ja alalla vallitsevia trendejä käsitellään yleisesti ja niitä lähestytään ohjelmoinnin kannalta. Tutkielmassa tutustutaan yleisiin seikkoihin, jotka aiheuttavat ohjelmistoissa niiden energiankulutuksen kasvua ja tutkitaan menetelmiä, joilla niiltä voidaan välttyä. Tutkimusmenetelmänä on kirjallisuuskatsaus. Tutkielmassa todettiin, että ohjelmistojen energiankulutuksen kasvu johtuu pitkälti laitteiden tehokkuuden kasvusta pisteeseen, jossa ohjelmistojen tehokkuuden optimoiminen ei aina ole välttämätöntä ohjelmiston käyttökokemuksen kannalta. Tutkielman tarkoituksena on myös antaa ohjelmoijalle konkreettisia työkaluja työnä energiatehokkuuden parantamiseksi perusteellisella suunnittelulla ja välttämällä yleisiä virheitä.

Asiasanat: Ohjelmistokehitys, vihreä ohjelmointi, energiatehokkuus, ICT-ala

Sisällys

1 Johdanto

Nykyinen yhteiskunta olisi täysin tunnistamaton ilman tietokoneita ja niiden tuomaa laskentatehoa. Maailman laskentakapasiteetin on ennustettu kasvavan miljoonakertaiseksi seuraavan kahden vuosikymmenen aikana ja sen seurauksena sen energiankulutus kaksinkertaistuu joka kolmas vuosi. Nykyisestä ilmasto- ja energiakriisistä johtuen, ovat menetelmät enegiankulutuksen ja hiilijalanjäljen vähentämiseksi välttämättömiä. [1] ICT (Information and Communcation Technology) sektorin arvioidaan kuluttavan noin 4-6% maailman energiankulutuksesta. [2]

Energiankulutus on myös merkittävä osa suurteholaskenta (High-Performance Computing) laitosten ylläpitokustannuksista, joten sen pienentäminen on kannattavaa paitsi ympäristön näkökulmasta, myös taloudelliset hyödyt ovat merkittäviä. [1]

Vihreä ohjelmointi (Green Coding) tarkoittaa ohjelmistokehityksen toimintamalleja ja tapoja, joilla optimoidaan luodun ohjelmakoodin aiheuttamaa energiankulutusta. Tällaisia tapoja ovat esimerkiksi vihreän ohjelmointikielen valitseminen, energian kulutuksen seuraaminen ja datan hallinnan optimoiminen. Näitä menetelmiä hyödyntämällä ohjelmoija voi siis itse suoraan vaikuttaa työnsä ympäristölle aiheuttamaansa kuormaan. [1] Kestävän kehityksen kannalta on tärkeää, että paitsi ohjelmoijat myös ohjelmistoalan yritykset ottavat vihreän ohjelmoinnin osaksi toimintaansa.

Tämän tutkielman tarkoituksena on tutkia erilaisia vihreän ohjelmoinnin menetelmiä ja selvittää miten ohjelmointi vaikuttaa energian kulutukseen. Tutkielma pyrkii vastaamaan seuraaviin tutkimuskysymyksiin:

TK1. Miten ohjelmointi vaikuttaa energiankulutukseen?

TK2. Mitä vihreän ohjelmoinnin tapoja ohjelmoija voi käyttää työnsä energiankulutuksen vähentämiseksi?

Ensimmäinen tutkimuskysymys keskittyy ICT-alan energiankulutukseen ja erityisesti ohjelmakoodiin vaikutuksesta siiheen. Kysymykseen pyritään vastaamaan tutkimalla eri ohjelmointikielien energiankulutusta sekä tutustumalla ICT-alalla vallitseviin käytänteisiin ja trendeihin, jotka ovat energiankulutuksen kannalta merkityksellisiä. Toisen tutkimuskysymyksen puolestaan on tarkoitus tutkia ohjelmoijan kannalta oleellisia menetelmiä ja työskentelytapoja, joilla hän voi vaikuttaa työnsä energiankulutukseen.

Tutkielma suoritettiin kirjallisuuskatsauksena hakemalla olennaisia artikkeleita ja kirjoja Volter-kannasta ja käyttämällä Google Scholar:ia. Hakusanojen kieleksi valittiin englanti, koska valtaosa alan kirjallisuudesta on kirjoitettu englanniksi. Lopulliset tiedonhaut suoritettiin seuraavan kuvaajan mukaisesti:

Kuva 1.1: Tutkielman hakusanat

Query			
“Programming” OR “ICT”	“green coding” OR “green ICT”	“AI” OR “artificial intelligence” OR “machine learning”	“technology”
AND			
“energy consumption”	-	“energy consumption” OR “novelty effect”	“novelty effect”

“Programming” ja “ICT” hakusanoilla haettiin tietoa ICT-alasta ja sen energian kulutuksesta. Vihreän ohjelmoinnin menetelmiä ja toimintatapoja haettiin hakusanoilla “green coding” ja “green ICT”. Tekoälyyn liittyvillä hakusanoilla etsittiin kirjallisuutta tekoälyn energiankulutuksesta ja “novelty effect” hakusanalla haettiin kirjallisuutta uutuudenviehätys ilmiöstä tekoälyyn liittyen sekä teknologia-alalla yleisesti.

Kuvaajan mukaisten tiedonhakujen tuloksia karsittiin lähinnä niiden tarkkuuden osalta. Tutkielma tarkastelee vihreää ohjelmointia yleisellä tasolla, joten tarkasti tiettyihin teknologioihin keskittyvä kirjallisuus karsittiin usein pois. Tekoälyn energiankulutukseen liittyvää kirjallisuutta kuitenkin käytettiin tutkielmassa, sillä se oli tutkielman kannalta oleellista.

Luvussa 2 käsitellään ICT-alan energiankulutusta ensin yleisellä tasolla ja myöhemmin tutkitaan ohjelmakoodin vaikutusta siihen. Luvussa käydään myös läpi alan muutosta viime vuosikymmenien aikana ja tutustutaan alalla vallitseviin trendeihin. Luvussa 3 pureudutaan vihreisiin menetelmiin tutustumalla ensin yleisiin virheisiin ja asioihin, jotka tekevät ohjelmakoodista energiatehotonta. Tämän jälkeen luvussa käydään läpi menetelmiä, joilla näitä virheitä voidaan välttää. Myöhemmin tarkastellaan

myös energiatehokkuutta lisäävien muutosten kannattavuutta ja tutustutaan uusien teknologioiden aiheuttamaan uutuuden viehätys ilmiöön ja sen merkitykseen ICT-alan energiankulutuksessa.

2 Tieto- ja viestintäteknologian energiankulutus

Energiantuotanto on välttämätöntä kaikille maailman valtioille niiden elintason ja talouden ylläpitämiseksi ja nostamiseksi ja maailma energiankulutus on kasvanut eksponentiaalisesti viime vuosikymmenien aikana. Vaikka maailmalla ollaan herätty ilmastonmuutoksen uhkaan ja toimia sitä vastaan on aloitettu erilaisten energiapolitiikan toimenpiteiden kautta, tuotetaan valtaosa maailman energiasta edelleen fossiilisilla polttoaineilla joiden päästöt ja rajallinen määrä johtavat suuriin ongelmiin lähitulevaisuudessa. Tässä luvussa tutustutaan tieto- ja viestintäteknologian energiankulutukseen yleisesti ICT-sektorin kannalta, sekä itse ohjelmakoodin vaikutuksista ohjelmistojen energiankulutukseen. Luvussa avataan myös ohjelmistoalalla vallitsevia haitallisia trendejä. [12]

2.1 ICT sektorin energiankulutus

ICT laitteet ja palvelut ovat suuresti muuttaneet tapoja joilla ihmiset työskentelevät, matkustavat ja viihdyttävät itseään. Kasvava osa ihmisistä ansaitsee elantonsa työskentelemällä tietokoneen edessä, monet autot käyttävät maailmanlaajuista paikallistamisjärjestelmää (GPS) ja videopelit sekä sosiaalinen media ovat tulleet osaksi monen arkipäivää. ICT sektorin kasvaessa, kasvavat myös sen energiankulutus ja muut ympäristöä kuormittavat vaikutukset.

ICT sektorin ympäritöjalanjäljelle ei ole olemassa vain yhtä mittalukua, vaan se näkyy useassa eri muodossa esimerkiksi resurssien louhiminen laitteiden tarvitsemien komponenttien tekemiseen ja siitä aiheutuva energiankulutus ja hiilidioksidipäästöt sekä vanhojen tai rikkinäisten laitteiden hävittäminen, varsinkin jos se tehdään huolimattomasti. [3] Tästä johtuen on olemassa useita eri menetelmiä ICT sektorin ympäristövaikutusten mittaamiselle riippuen siitä mitä halutaanko tarkastella tuotteita, palveluita, organisaatioita ja projekteja tai paikkoja kuten kaupunkeja. Nämä menetelmät taas voidaan karkeasti jakaa elinkaareen perustuviin menetelmiin ja organisaation hiilijalanjäljen laskemismenetelmiin. [4]

Tämän tutkielman kannalta oleellista on itse laitteiden ja erilaisten ICT palveluiden käytöstä aiheutuva energiankulutus. ICT tuotteiden elinkaaresta juuri käyttövaiheen on todettu aiheuttavan merkittävä osa koko tuotteen elinkaaren aina valmistuksesta hävittämiseen aiheutuvasta kuormasta ympäristölle. [3]

2.2 Ohjelmiston energiankulutus

Laitteiston ja niihin liittyvät edellä mainitut ympäristövaikutukset ovat ICT sektorin laajalti tunnustamia jonka ansiosta niiden parissa työskentelevät tahot voivat ottaa ne huomioon tuotteita suunnitellessa. Itse ohjelmiston rooli energiankulutuksen vähentämisessä on jäänyt alalla vähemmälle huomiolle, vaikka ohjelmistokehityksen tutkijat ovat vakuuttuneita sen merkityksestä. Koska ohjelmistoja tuottavilla tahoilla on rajallinen käsitys ohjelmiston energiankulutuksen vähentämisestä, on näillä tahoilla heikommat mahdollisuudet vaikuttaa tuotteidensa energiankulutukseen ja täten ohjelmiston osuus erilaisissa energiansäästö suunnitelmissa jää usein pieneksi. Tästä johtuen alan toimijat eivät välttämättä kykene täyttämään asiakkaiden vaatimuksia tai yritysten kestävyys (Corporate Sustainability) tavoitteita. [5]

Yksinkertaisimmillaan ohjelmistojen energiankulutus aiheutuu tiedonkäsittelystä ja sen siirrosta. Tiedonkäsittely vaati prosessorilta kellosyklejä ja sen siirtäminen vaatii sekä prosessori- että I/O-tehoa (Input Output). Tämä yksinkertaisuus kuitenkin häviää nopeasti, kun otetaan huomioon ohjelmistojen monimutkaisuus, kerroksellisuus ja mittauspisteiden puute. Yhdessä laitteessa voi olla käynnissä satoja prosesseja samanaikaisesti ja jokainen on arkkitehtuuriltaan, algoritmeiltaan ja muilta ominaisuuksiltaan erilainen. Tästä johtuen, vaikka laitteen kuluttamaa energiaa voidaan mitata sen kuluttaman sähkötehon määrästä, on se miten se jakautuu eri osien kesken usein epäselvää ja vaikeasti mitattavaa. Ongelma moninkertaistuu, kun käytössä onkin useita laitteita kuten päätelaite ja erilaisia palvelimia ja jokaisessa on omat erilaiset ohjelmistonsa. Useasta lähteestä hankittua mittausdataa ei myöskään usein voi suoraan laskea yhteen, sillä tapa mitata, tulkita mittauksia tai niiden sisältämät asiat voivat olla lähteiden välillä erilaisia. [10]

ICT laitteiden teho on kasvanut nopeasti ja myös niiden energiatehokkuus on kaksinkertaistunut noin puolentoista vuoden välein. Tästä huolimatta, ICT sektorin energiankulutus jatkaa kasvuaan. Osatekijänä tässä ovat itse ohjelmiston tuottajat. Ohjelmointi on muuttunut merkittävästi viimevuosikymmenien aikana. 1980- sekä 1990-luvuilla ohjelmointi oli huomattavasti lähempänä laitetasoa eli itse tietokonetta, mikä puolestaan tarkoitti, että ohjelmoijalla piti olla parempi ymmärrys tietokoneen arkkitehtuurista. Ohjelmointi oli hidasta ja virheitä oli helppo tehdä. Nykyisin ohjelmointi tapahtuu usein paljon korkeammalla abstraktion tasolla. Valittu ohjelmointikieli ja kielen kääntäjä voivat hoitaa ohjelmoijan puolesta monia tehtäviä,

jotka hänen olisi aiemmin pitänyt tehdä itse. Tämä on nopeuttanut ohjelmointia huomattavasti ja vähentänyt siihen liittyviä rahallisia kustannuksia. Kehityssuunnan hintana on uusien ohjelmointikielien aiheuttama suurempi energiankulutus. [6]

Kuva 2.1: Eri ohjelmointikielien vertailu C kieleen [8]

Total

Language	Energy	Language	Time	Language	Mb
C	1.00	C	1.00	C	1.00
Rust	1.03	Rust	1.04	Rust	1.05
C++	1.34	C++	1.56	C++	1.17
Ada	1.70	Ada	1.85	Ada	1.24
Java	1.98	Java	1.89	Java	1.34
Pascal	2.14	Pascal	2.14	Pascal	1.47
Chapel	2.18	Chapel	2.83	Chapel	1.54
Lisp	2.27	Lisp	3.02	Lisp	1.92
Ocaml	2.40	Ocaml	3.09	Ocaml	2.45
Fortran	2.52	Fortran	3.14	Fortran	2.57
Swift	2.79	Swift	3.40	Swift	2.71
Haskell	3.10	Haskell	3.55	Haskell	2.80
C#	3.14	C#	4.20	C#	2.82
Go	3.23	Go	4.20	Go	2.85
Dart	3.83	Dart	6.30	Dart	3.34
F#	4.13	F#	6.52	F#	3.52
JavaScript	4.45	JavaScript	6.67	JavaScript	3.97
Racket	7.91	Racket	11.27	Racket	4.00
TypeScript	21.50	TypeScript	26.99	TypeScript	4.25
Hack	24.02	Hack	27.64	Hack	4.59
PHP	29.30	PHP	36.71	PHP	4.69
Erlang	42.23	Erlang	43.44	Erlang	6.01
Lua	45.98	Lua	46.20	Lua	6.62
Jruby	46.54	Jruby	59.34	Jruby	6.72
Ruby	69.91	Ruby	65.79	Ruby	7.20
Python	75.88	Python	71.90	Python	8.64
Perl	79.58	Perl	82.91	Perl	19.84

Prereira et al. [7] analysoi eri ohjelmointikielien suoritusaikaa, muistin käyttöä ja energiankulutusta. He käyttivät kymmentä eri ohjelmointiongelmaa analyysissaan. Energiankulutukseltaan tehokkain oli C ohjelmointikieli, joka on laitetasoa lähellä oleva kieli ja vaatii ohjelmoijalta merkittävää ammattitaitoa. C ohjelmointikieltä käytettiin vertailukohteena muille kielille. Vertailusta selviää, että mikäli kaikki ohjelmisto

maailmassa olisi tehty C:llä, olisivat huolet niiden energiatehokkuudesta vähäisempiä. Mikäli vertaamme C:tä suosittuun ohjelmointikieleen Python:iin, voimme kuvaajan perusteella todeta Python:in vaativan lähes 76 kertaa enemmän energiaa, kuin jos sama tehtävä suoritettaisiin C:llä. JavaScript:lla ero olisi nelinkertainen ja C#:lla kolminkertainen. Myös monet muut kielet maailman suosituimpien ohjelmointikielien listalta kuluttavat energiaa moninkertaisesti C:en verrattuna. Energiankulutusta onkin siis mahdollista vähentää huomattavasti valitsemalla tehtävään sopiva, vähemmän energiaa kuluttava ohjelmointikieli. [7] [6]

Kuva 2.2: Kymmen suosituinta ohjelmointikieltä[7]

Rank	Language	Share	1-year trend
1	Python	29.56 %	+1.6 %
2	Java	15.66 %	-0.2 %
3	JavaScript	8.16 %	-1.0 %
4	C/C++	6.76 %	-0.0 %
5	C#	6.58 %	-0.1 %
6	R	4.64 %	+0.2 %
7	PHP	4.20 %	-0.7 %
8	TypeScript	2.95 %	-0.0 %
9	Swift	2.64 %	-0.1 %
10	Rust	2.55 %	+0.5 %

Toinen ohjelmistojen energiatehokkuuteen vaikuttava asia on ohjelmiston kasvu. Niklaus Wirth tutki tätä ilmiötä jo vuonna 1995 [9] ja hänen analyysiaan aiheesta kutsutaan usein Wirthin laiksi, jonka mukaan ohjelmistot muuttuvat hitaammiksi nopeammin, kuin laitteet kehittyvät nopeammiksi. Tämä tarkoittaa, että laitteiden kehittyminen ei pysy ohjelmistojen kehittymisen perässä. [6][9] Wirthin lain esimerkkinä voidaan pitää niin kutsuttua ohjelmiston paisumista (Software Bloat), jossa kehitettävän ohjelmiston uudet versiot vievät yhä enemmän muistia, laskentatehoa, levytilaa tai ne vaativat parempia ja tehokkaampia laitteita ja samalla tarjoavat ohjelmiston käyttöä nähden vain pieniä parannuksia tai vastaavasti ohjelmistossa on

liikaa ominaisuuksia sen käyttötarkoitukseen (Feature Creep). Tämä kaikki johtaa lopulta energiankulutuksen kasvuun. [6]

2.3 ICT-sektorin trendit

ICT-alan vallitsevat trendit ovat viemässä alaa energiankulutuksen ja ilmastonmuutoksen kannalta väärään suuntaan. Datan määrä kasvaa räjähdysmäisesti ja se kumuloituu, kun vanhaa dataa ei tuhota uuden alta. Ohjelmistoja mitataan usein ominaisuuksien pohjalta eikä energiankulutuksen kannalta ja niiden tehokkuus on toissijaista kunhan se ei merkittävästi vaikuta käyttökokemukseen. ICT-alalla kaivataan jatkuvasti uusia ja tehokkaampia laitteita, mikä puolestaan mahdollistaa entistä tehottomampien ohjelmistojen luomisen. Ongelmaa pahentaa entisestään vanhojen ohjelmistojen ohjelmistotuen loppuminen. Toisin sanoen vanhat laitteet eivät ajan mittaa muutu käyttökelvottomiksi, vaan trendin myötä niistä tehdään käyttökelvottomia, koska ne eivät enää täytä uusille laitteille suunniteltujen ohjelmistojen vaatimuksia. [10]

Mobiilijärjestelmien suosio on kasvanut huomattavasti. Kasvava osa datasta siirretään langattomasti mobiiliverkkojen avulla ja mobiililaitteella hoidetaan enemmän asioita. Tämä vaikuttaa energiankulutukseen, koska tiedonsiirto on merkittävästi energiatehottomampaa mobiiliverkkojen välityksellä, kuin se olisi langallisella verkkoyhteydellä. Mobiililaitteille tehdyt sovellukset ovat myös usein ilmaisia ja niiden rahoitus katetaan esimerkiksi mainoksilla tai sovellusten sisäisillä ostoksilla. Erityisesti mainosverkostojen tekemät automatisoidut toiminnot kuten mainospaikkojen huutokaupat kuluttavat huomattavasti energiaa. Aalto-yliopiston tekemässä tutkimuksessa mainosverkostojen arvioitiin vievän noin 10% internetin energiankulutuksesta. [11] Trendissä on kuitenkin myös hyviä puolia. Mobiililaitteet ovat tietokoneita energiatehokkaampia datanprosessoinnissa, koska mobiililaitteilla energiankulutus on käyttökokemuksen kannalta oleellisempaa kuin perinteisillä tietokoneilla, joten nämä laitteet on suunniteltu energiaa säästäviksi. Mobiililaitteiden käyttäminen voi siis olla järkevää, kunhan sillä vältetään tietokoneen käyttämistä. [10]

Tekoälyn käytön kasvu on yksi uusimmista ICT-alan trendeistä. Tekoäly on muuttunut abstraktista tieteisfiktion käsitteestä jokapäiväiseksi työkaluksi, joka on helposti ihmisten saatavilla ja tekoälyyn ja koneoppimiseen pohjautuvia ratkaisuja syntyy jatkuvasti lisää. Näiden ratkaisujen taustalla on valtavat datamäärät, jotka

puolestaan vaativat merkittävästi laskentatehoa ja täten energiaa. Vielä ei tiedetä, tulevatko tekoälyn tarjoamat hyödyt kattamaan siitä syntyvän energiankulutuksen. [10]

Trendien suuntana on siis jatkuva ohjelmistojen ja datan määrän kasvaminen ja monimutkaistuminen ja tämä vaatii aina nopeampia laitteita ja verkkoja. Nämä puolestaan mahdollistavat entistäkin tehottomammat ohjelmistot ja suuremmat datamäräät. Ala on joutunut kierteeseen, jossa laskentehon tarpeeseen vastataan hankkimalla sitä aina lisää sen sijaan, että ohjelmistoja optimoitaisiin paremmin ja olemmassaolevasta laskentatehosta saataisiin enemmän irti. ICT-sektorin energiankulutuksen vähentämiseksi tämä kierre pitäisi uskaltaa katkaista. [10]

3 Vihreät menetelmät

Vihreät menetelmät ovat ohjelmistokehityksen toimintamalleja, joilla pyritään optimoimaan ohjelmakoodin energiankulutusta. [1] Niillä pyritään siirtymään pois ohjelmistoalalla vallitsevasta käsityksestä, jonka mukaan tehoa on loputtomasti saatavilla ja ohjelmistojen ei tarvitse toimia vanhemmilla laitteilla. Tämä käsitys johtaa energiatehottomiin ohjelmistoihin, jotka eivät edes välttämättä toimi hyvin, sillä vihreät ohjelmat ovat usein myös tehokkaita ja hyvin suunniteltuja ohjelmia. Tässä luvussa pureudutaan tarkemmin siihen, mitä virheitä energiankulutuksen optimoinnin kannalta ohjelmoinnissa usein tehdään joko yleisessä ohjelmiston suunnittelussa ja myös itse ohjelmakoodintasolla. Luvussa käydään myös läpi keinoja, joilla ohjelmoija voi itse vaikuttaa työnsä energiankulutukseen [10]

3.1 Tehottomat ohjelmistot

Kuten aikaisemmin jo todettiin, ohjelmointi oli ennen huomattavasti lähempänä laitetasoa ja tietokoneet olivat nykyisiin verrattuna tehottomia. Tämän johdosta ohjelmien piti olla tehokkaita, sillä muuten niitä ei olisi voitu toteuttaa. Koska nykyään laitteissa on tehokkuutta saatavilla normaaliin arkikäyttöön käytännössä rajattomasti, voi sitä niin sanotusti hukata sen juuri vaikuttamatta ohjelmiston käyttöön tai toimintaan. Energiatehokkaiden ja -tehottomien ohjelmistojen kontekstissa hukalla tarkoitetaan ylimääräisiä toimintoja, jotka turhaan kuluttavat energiaa. Käydään seuraavaksi läpi muutamia hukkia: [10]

- Turhat ohjelmistot: Jos ohjelmisto on turha, on kaikki sen käyttämä sen energia hukkaa. Tällainen ohjelmistoja voivat esimerkiksi olla tietokoneiden mukana tulevat sovellukset, joita loppukäyttäjä ei käytä. Turhia ohjelmistoja miettiessä pitää kuitenkin huomioida turhuuden olevan subjektiivinen käsite. Yhden käyttäjän turha sovellus voi toisella olla päivittäisessä käytössä.
- Väärä käyttötarkoitus: Ohjelmistoja voidaan käyttää väärään tarkoitukseen. Vaikka ne voivat siitä suoriutua käyttökokemuksen puolesta riittävän hyvin, niitä ei ole suunniteltu siihen, jolloin energiankulutus usein kasvaa. Tämä voi johtua esimerkiksi siitä, että väärän käyttötarkoituksen takia joudutaan tekemään paljo raskaita tiedonhakuja.
- Käyttäjien virheet: Käyttäjien tekemät virheet, kuten hiirellä väärää kohtaa painaminen, aiheuttavat ohjelmistossa toimintoja, jotka ovat hukkaa. Näistä

virheistä on liki mahdotonta päästä kokonaan eroon, mutta esimerkiksi hämmentävät käyttöliittymät lisäävät virheiden mahdollisuutta. Tähän liittyvät oleellisesti myös pimeät käytännöt (engl. dark patterns). Maksullinen suoratoistopalvelu voi esimerkiksi tarkoituksella piilottaa tilauksen peruutus toiminnon tai tehdä siitä muuten haasteellista. Tämä aiheuttaa käyttäjältä turhia toimintoja, jotka ohjelmisto kuitenkin käsittelee.

- Väärä arkkitehtuuri: Ohjelmiston arkkitehtuuri määrittää sen, mikä ohjelmakoodille on helppoa ja mikä haastavaa. Jos valittu arkkitehtuuri on vääränlainen, joudutaan usein toiminnallisuudet toteuttamaan monimutkaisesti. Tämä lisää paitsi energiankulutusta, myös virheiden määrä kasvaa.
- Väärä tietomalli ja turha tai optimoimaton tieto: Tietomallin ollessa käyttötarkoitukseen nähden väärä, ohjelma joutuu joutuu jatkuvasti etsimään ja yhdistelemään tietueita, jotta ne voidaa näyttää käyttäjälle. Turha tieto taas kasvattaa hukkaa, koska sen siirtäminen ja käsitteleminen on yksinkertaisesti tarpeetonta. Optimoimaton tieto puolestaan vaatii turhaa energiaa, koska sen käsitteleminen on työläämpää.
- Algoritmien tehottomuus: Vaikka vakiintuneiden ohjelmointikielien ja kirjastojen algoritmit ovat usein hyvin optimoituja, pätee se vain silloin, kun niitä käytetään oikeisiin käyttötarkoituksiin. Vastaavasti väärin valittu tai itse tehty algoritmi saattaa tehokkuudeltaan olla erittäin heikko.
- Kirjastot: Ohjelmistoja luodaan usein käyttämällä monia kirjastoja, joiden toiminnallisuuksia niissä käytetään. Kirjastot saattavat myös käyttää muita kirjastoja. Pahimmillaan voi ohjelmiston ohjelmakoodista omaa koodia olla vain muutama prosentti verrattuna kirjastojen ohjelmakoodiin. Kuitenkin näiden kirjastojen koodista saatetaan varsinaisesti käyttää vain murto-osaa. Tämä muodostaa ongelman ohjelmistoa siirrettäessä laitteelle. Voi käydä niin, että sama kirjasto on laitteella useita kertoja eri ohjelmistojen sisällä.

3.2 Vihreä ohjelmoija

Vihreän ohjelmoinnin toimintamallit auttavat ohjelmoijaa ja ohjelmistoyrityksiä vähentämään tekemiensä ohjelmistojen energiankulutusta. Näillä toimintamalleilla ja

menetelmillä pyritään vähentämään ohjelmistojen hukkaa ja optimoimaan niiden toimintaa. Millainen sitten on vihreän ohjelmoijan työkalupakki? [1][10]

Minimointi on yksi tapa vaikuttaa ohjelmistojen energiankulutukseen. Minimointi ei suoranaisesti pyri hukkien tunnistamiseen ja korjaamiseen, vaan nimensä mukaisesti se pyrkii karsimaan ohjelmistoista tarpeettomia asioita pois ja näin laatimaan mahdollisimman kompakteja ratkaisuja, jotka kuitenkin täyttävät ohjelmiston tarpeet. Perusteellinen suunnittelu on minimoinnin kannalta välttämätöntä. Jokaisen ominaisuuden tarpeellisuus pitää kyseenalaistaa ja miettiä, voidaanko sitä ilman pärjätä. Pitää kuitenkin huolehtia, että tärkeät ominaisuudet toteutetaan tai muuten tuloksena on puolivalmis ohjelmisto, jota joudutaan korjaamaan jälkeenpäin. [10]

Tiedon käsittely ja tallentaminen on erittäin olennainen osa ohjelmistojen ohjelmistojen energiankulutusta, energiankulutusta, joten sen minimoiminen on kannattavaa. Ohjelmistojen prosessit pitää laatia siten, että ne mahdollistavat tarpeettoman tai vanhentuneen tiedon helpon poistamisen ja vain prosessien lopputulokset tallennetaan ja välitulokset tuhotaan. Mikäli tiedon pitää olla saatavilla, mutta sitä ei tarvita ohjelmiston jokapäiväisessä toiminnassa, voidaan se niin sanotusti kylmävarastoida erilaisiin pilvipalveluihin. Videoiden ja kuvien resoluution ja pakkauksen säätämisellä hyvin voidaan niiden käsittelemisestä tehdä tehokkaampaa. Vastaavanlaista minimointia kannattaa tehdä myös käyttäjän itse ohjelmistoon syöttämälle sisällölle. Ohjelmistojen usein keräämään analytiikkatiedon tarpeellisuutta pitää miettiä. Osa tiedosta on varmasti tarpeellista ohjelmiston käyttäjäkokemuksen parantamiseksi ja liiketoiminnan ylläpitämiseksi, mutta turhan tiedon keräämistä on syytä välttää. [10]

Siirretyn tiedon minimoiminen liittyy läheisesti tiedon minimointiin. Mitä vähemmän sitä käsitellään, sitä vähemmän sitä myös siirretään. Tiedon siirtämisen energiatehokkuutta on kuitenkin mahdollista parantaa myös muilla tavoilla. Kommunikoinnin välit kannattaa säätää mahdollisimman pitkiksi, sillä se vähentää suoraan tiedon siirtämistä. Ohjelmiston toimintaa ja käyttäjäkokemusta ei kuitenkaan pidä vaarantaa. Pakkaamalla tietoa, sen kokoa voidaan pienentään, jolloin sen siirtämisen energiakustannukset ovat pienemmät. Myöskin protokollan tai viestimuodon valinnalla on merkitystä. Osa niistä selviää pienemmillä datamäärillä kuin toiset. Ohjelmiston tilan päivittämiseen on riittävää siirtää ainoastaan muuttunut tieto ja tiedon siirron keskittäminen tähän tietoon vähentään tiedon siirtoa ja näin energiankulutusta. Tämän kannalta oleellista on muuttuneen tiedon tunnistaminen. Ohjelmisto voi

esimerkiksi lähettää palvelimelle tiedon muuttuneesta tiedosta muun kommunikoinnin yhteydessä. HTTP-otasketietojen minimoinnilla ja HTTP-uudelleenohjausten vähentämisellä voidaan vähentää ohjelmiston toiminnan kannalta turhia HTTP-pyyntöjä ja parantaa kommunikoinnin energiatehokkuutta. [10]

Ohjelmakoodin määrää vähentämällä saadaan pienennettyä ohjelmiston kokoa, jolloin sen siirtäminen verkon kautta on nopeampaa ja energiatehokkaampaa ja myös sen käynnistyminen saattaa nopeutua. Jos ohjelmistossa on osia, joita ei enää käytetä, voi niihin liittyvän ohjelmakoodin yksinkertaisesti poistaa ja mikäli sitä tarvitaan joskus tulevaisuudessa, voidaan se löytää versionhallintajärjestelmästä. Vaikka versionhallinta itsessään lisää energiankulutusta, lisää se kehittämisen luotettavuutta ja vähentää virheitä merkittävästi. Aikaisemmin totesimme ohjelmistoissa käytettävän nykyisin paljon kirjastoja ja kuinka niistä saatetaan varsinaisesti käyttää vain pientä osaa, mikä tarkoittaa suuren osan kirjastojen ohjelmakoodista olevan turhaa. Tällaisissa tilanteissa on kannattavaa miettiä, voidaanko kirjaston pieni käytetty osa kopioida lisenssiehtojen sen salliessa tai kirjoittaa se itse. Kirjastoista on joskus olemassa myös kevyempiä versioita, jotka toteuttavat valtaosan sen toiminnoista pienemmällä ohjelmakoodin määrällä. Tällöin voidaan kirjasto vaihtaa tehokkaampaan vaihtoehtoon. Ohjelmistosta voidaan myös karsia ominaisuuksia. Jos jotakin ominaisuutta ei käytetä usein tai se on vanhentunut, kannattaa se poistaa kokonaan. [10]

Ohjelmistojen energiankulutus ja niiden suoritusaika ovat vahvasti kytköksissä toisiinsa, joten ohjelmiston tehokkuuden nostaminen parantaa myös sen energiatehokkuutta. Algoritmit ovat usein olennainen tekijä ohjelmistojen tehokkuudessa. Niiden välillä on kuitenkin eroja ja kiireellisten aikataulujen takia kehittäjä saattaa ottaa käyttöön ensimmäisen vastaan tulevan toimivan algoritmin ilman, että sen tehokuutta on mietty riittävästi. Valittujen algoritmien tehottomuutta voi olla hankala huomata kehitysvaiheessa niille syötetyn testidatan pienuuden ja kehittäjän laitteiden tehokkuuden takia. Onkin suositeltavaa käyttää algoritmeja, joiden tehokkuudesta löytyy riittävästi tutkimusta. Tietorakenteen valinnalla on myös suuresti vaikutusta ohjelmiston toimintaan ja tehokkuutteen. Osa niistä on tehokkaita suurillakin tietomäärillä ja toiset on suunniteltu rajallisemmille tietomäärille ja mikäli ohjelmiston tiedonhakua ei ole toteutettu hyvin, menetetään tehokkaankin tietorakenteen hyödyt nopeasti. [10]

Kun aikaisemmin käsiteltiin ohjelmistojen energiankulutusta, todettiin ohjelmointikielellä itsellään olevan siihen merkittävä vaikutus. Ohjelmointikielen

valinnassa sen energiatehokkuuden tulisi olla yksi kriteereistä. Usein ohjelmointikieltä ei kuitenkaan voida valita yksin tämän perusteella, vaan muista tekijöistä johtuen päädytään energiatehottomampaan ohjelmointikieleen. Tällaisessa tilanteessa voidaan pohtia ohjelmiston osittaista toteuttamista energiatehokkaammalla ohjelmointikielellä. Mikäli ohjelmistossa on toimintoja, jotka vaativat muita merkittävästi suurempaa laskentatehoa, voidaan nämä toiminnot toteuttaa esimerkiksi C:llä. Ohjelmointikielien yhdisteleminen ei kuitenkaan ole ohjelmointikielen valinnan hopealuoti, sillä joitakin ohjelmointikieliä on hankalaa yhdistää, mutta ratkaisu on silti hyvä vaihtoehto erityisesti refaktoroinissa, eli ohjelmistojen osittaisessa uudelleen kirjoittamisessa. Ohjelmointikielien energiatehokkuuteen liittyy myös niiden ajoympäristö. Tulkattavien ja tavukoodiski käännettävien ohjelmointikielien ajoympäristöjen välillä on usein tehokkuuseroja ja vaikka erot usein ovat eri ajoympäristöjen välisiä, voi ajoympäristön version päivittäminen uudempaan lisätä tehokkuutta. Onkin kannattavaa seurata ajoympäristöjen kehittymistä ja valita niistä ohjelmistolle parhaiten sopiva. [1] [10]

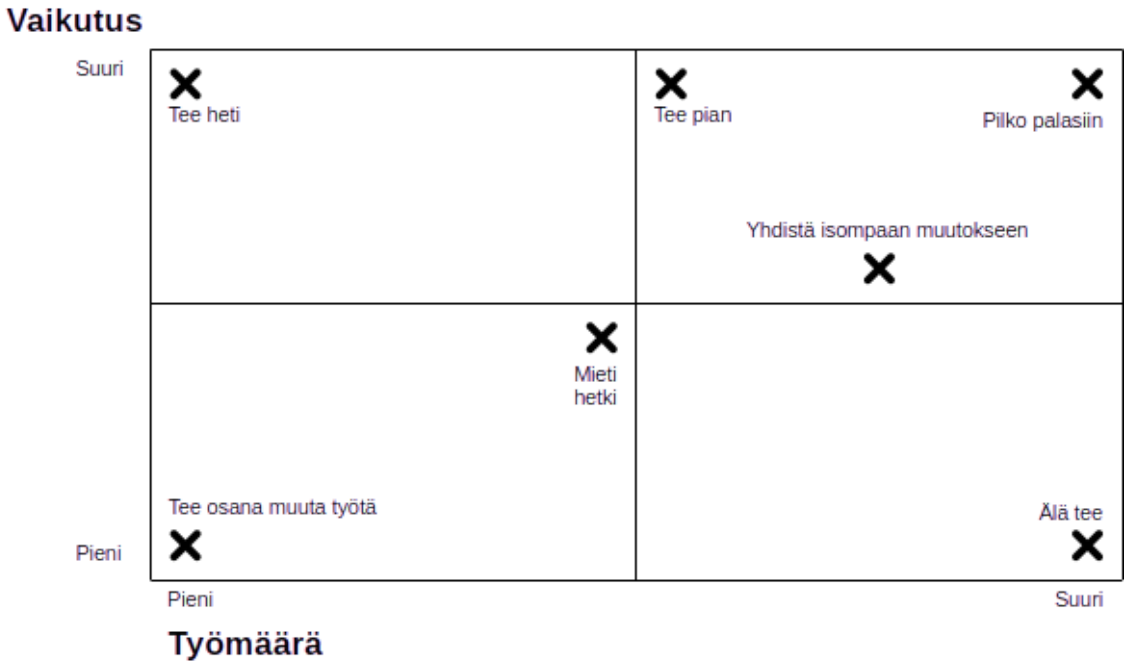
Työpöytä- ja selainsovellukset ovat usein taustalla käynnissä käyttäjän tehdessä laitteella muita asioita. Käyttöjärjestelmät ja selaimet usein säätävät automaattisesti toimintaansa tällaista tausta-ajoa varten kaventamalla tausta-ajettaville toiminnoille allokoituja resursseja. Mikäli on saatavilla signaali tausta-ajoon siirtymisesestä, se voidaan huomioida ohjelmistoja kehitettäessä. Ohjelmiston ollessa tausta-ajossa, voidaan käyttöliittymä jättää päivittämättä ja reaktioaikoja voidaan säätää pidemmiksi esimerkiksi hidastamalla tiedonhakua. [10]

3.3 Muutoksen kannattavuus

Edellisissä luvuissa käsiteltiin tavanomaisia ohjelmistojen energiatehokkuuden sudenkuoppia sekä keinoja, jotka huomioon ottamalla ohjelmoija voi niitä välttää. Nämä keinot eivät kuitenkaan usein ole ilmaisia, vaan ohjelmiston luonteesta riippuen ne vaativat vaihtelevia määriä työtunteja ja siten rahaa, jota harvoin on käytössä rajattomasti. Myös ohjelmiston käyttökokemus pitää huomioida, sillä vaikka energiatehokkuuden kannalta olisi mielekästä, että ohjelmistot olisivat niin paljaita ja hitaita kuin mahdollista, haittaisi tämä ohjelmiston käyttöä kohtuuttoman paljon. Energiatehokkuutta edesauttavia toimia harkittaessa pitääkin siis pitää mielessä toimien vaikutuksen suuruus ja niiden toteuttamisesta aiheutuvat kustannukset ja seuraukset. [10]

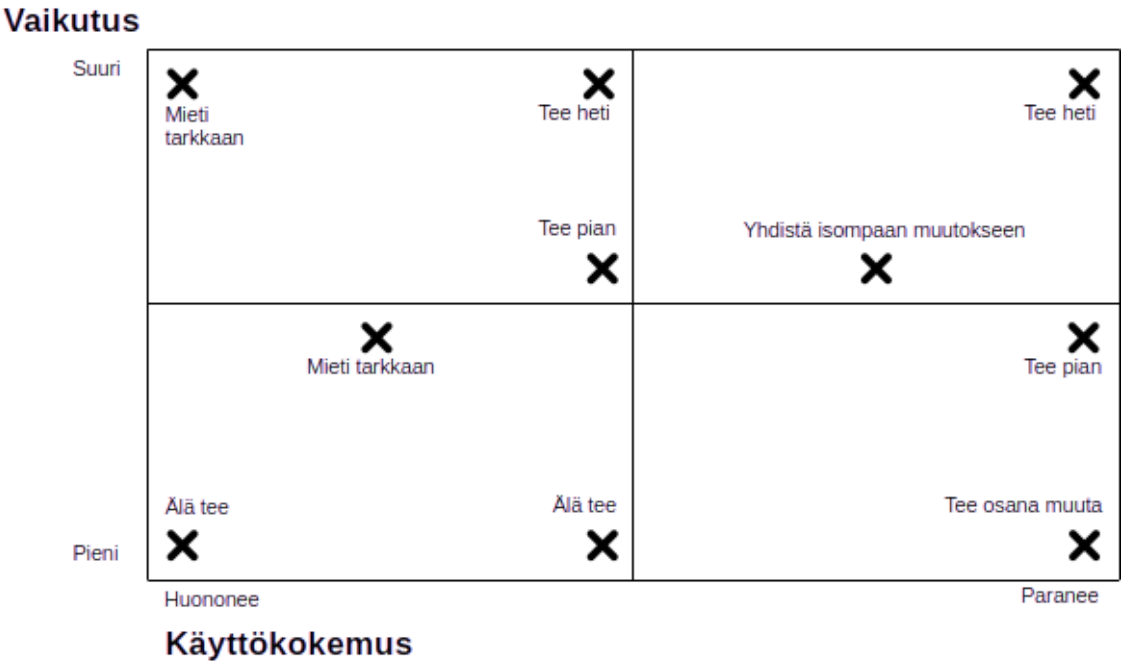
Mietittäessä toimia energiatehokkuuden parantamiseksi, yritysten pitää miettiä, mitä suuri tai pieni työmäärä tai vaikutus heille tarkoittaa. Yritysten tilanteissa ja tarpeissa on merkittäviä eroja joten yleispäteviä mittareita on hankala luoda. Kun nämä arviot on tehty, voidaan muutosten kannattavuutta arvioida yksinkertaisella nelikenttä kuvaajalla. [10]

Kuva 3.1: Vaikutus ja työmäärä nelikenttä [10]



Käyttökokemus on yhtälailla yksilöllistä eri yrityksille ja ohjelmistoille, sillä käyttökokemuksen kannalta tärkeät ominaisuudet vaihtelevat niiden välillä. Energiatehokkuuden parantamistoimien ei pitäisi kohtuuttomasti huonontaa näitä ominaisuuksia. Toimia, jotka näihin ominaisuuksiin kuitenkin vaikuttavat pitää miettiä ja suunnitella huolella, jotta ne voidaan toteuttaa tavoilla, jotka eivät merkittävästi huononna käyttökokemusta. Tämä ei kuitenkaan välttämättä aina ole mahdollista. Toimien kannattavuutta käyttökokemuksen kannalta voidaan myös havainnoida nelikenttä kuvaajalla, jossa työmäärään sijaan tarkastellaan vaikutusta käyttökokemukseen. [10]

Kuva 3.2: Vaikutus ja käyttökokemus nelikenttä [10]



3.4 Uutuudenviehätys

Uusien teknologioiden syntyessä tai niiden tullessa tunnetummiksi, liittyy niihin eräänlaista uutuuden viehätystä. Tämä tarkoittaa, että teknologian tuoma hyöty ja arvo koetaan suuremmiksi, kuin ne todellisuudessa saattavat olla yksinkertaisesti siksi, että se on uusi ja täten hieno teknologia. Ilmiö on läsnä, kun mietitään investointeja ja vaikuttaa näissä tehtäviin valintoihin. [14] Ilmiön haittana on uusien teknologioiden seurausten ja vaikutusten usein puutteellinen ymmärtäminen ja data esimerkiksi energiankulutuksen kannalta. Tämä voi johtaa siihen, että investointiin valitaan huonompi uusi teknologia, vaikka tarjolla olisi vanha ja seurauksiltaa paremmin ymmärretty vaihtoehto. [10]

3.4.1 Tekoäly

Tekoälyllä tarkoitetaan yleisesti ihmismäisen älykkyyden ja toimintakyvyn saavuttamista koneellisesti ja erityisesti tietokonejärjestelmillä. Käytännössä tämä tarkoittaa, että kone osaa ohjata ja muuttaa toimintaansa ilman ihmisen apua. Konseptina tekoäly ei ole uusi, mutta lähivuosina ChatGPT:n ja muiden helposti saatavilla olevien tekoäly työkalujen ansiosta teknologia on noussut jälleen esille ja voidaankin lähes puhua tekoälyn alkuinnostuksesta (AI hype).

AI innostus on esimerkki uutuuden viehätyksestä. Tekoälyn käyttäminen ongelmien ratkaisemiseksi nähdään trendikkäänä ja innovatiivisena, joten sitä sitä saatetaan käyttää myös tilanteissa, joissa sitä ei tarvita. Datan käsittelyssä voidaan esimerkiksi käyttää raskasta koneoppimismallia perinteisen ja kevyemmän algoritmin sijaan. [15]

Tekoälyn energiankulutus voidaan jakaa kolmeen osaan. Ensimmäinen on tekoälyn kouluttamiseen käytettävän materiaalin koostaminen ja muokkaaminen käytettävään muotoon, sillä opetusmateriaaliksi ei kelpaa mikä tahansa. Lisäksi opetusmateriaalia tarvitaan suuria määriä, joten sen kerääminen ja muokkaaminen on usein tarpeellista automatisoida, mikä lisää prosessin energiankulutusta. Toinen on tekoälyn opettamiseen kuluva energia, missä tekoäly koulutetaan koostetulla opetusmateriaalilla. Tämän koulutusvaiheen energiankulutukseen vaikuttavat useat eri tekijät, kuten valittu tekoälymalli ja opetusmateriaalin muoto. Videoiden tai kuvien käyttäminen materiaalissa kuluttaa enemmän energiaa kuin pelkän tekstin käsittely. Itse opetusmateriaalin tai koulutetun tekoälymallin koko eivät suoraan määritä koulutusvaiheen energiankulutusa, sillä eri tekoälymallien oppimisen energiatehokkuuksien välillä on eroja. Energiankulutuksen kannalta on myös olennaista se, että tekoäly pitää joskus kouluttaa uudelleen, jotta se ei toimisi vanhentuneen tiedon varassa. Viimeinen osio on itse tekoälyn käyttämisestä aiheutuva energiankulutus. Periaatteessa tekoälyn käytön energiankulutus aiheutuu samalla tavalla kuin muussakin ohjelmakoodissa, eli tiedon siirtämisestä ja käsittelemisestä, mutta tekoälymallit ovat usein monimutkaisia ja kooltaan isoja, jolloin tiedonsiirtoa ja -käsittelyä tapahtuu paljon. Käyttövaiheen energiankulutukseen voidaan myös laskea tekoälymallin lataaminen ja vastaavat käyttöönottoon liittyvät toimenpiteet. Tekoälymallien usein suuresta koosta johtuen käyttöönoton energiankulutus ei ole merkityksetöntä. [10]

Teknologiana tekoäly on energiaintensiivinen, joten sen käyttäminen ohjelmistoissa lisää niiden energiankulutusta. Tätä ei kuitenkaan pidä tulkita niin, että tekoäly ei kannata ikinä käyttää, vaan pitää yksinkertaisesti miettiä, onko se oikea ratkaisu ongelmaan vai halutaanko sitä käyttää vain siksi, että voidaan sanoa ohjelmistossa olevan uutta ja hienoa teknologiaa. Samanlaista asennetta on hyvä käyttää myös muiden uusien teknologioiden kanssa. Mikäli tekoälyn koetaan olevan sopiva ratkaisu, on syytä pohtia sen rajaamista, sopivan tekoälymallin valitsemista ja sen konfiguraatiota sekä mallin kouluttamista ja siihen käytettävää materiaalia. Rajaamalla tekoälyn käyttö ratkaisussa vain sitä vaativiin ominaisuuksiin vältytään sen turhalta käytöltä. Oikean mallin valitseminen on tärkeää, sillä näiden soveltuvuuksissa ja

tehokkuuksissa eri käyttötarkoituksiin on eroja. Koulutusmateriaalin taas pitää vastata kehitettävän ratkaisun tarpeita ja sen tuottaminen ja muokkaaminen tulee toteuttaa mahdollisimman energiatehokkaasti. Tekoälymalleissa on usein myös paljon niiden toimintaan vaikuttavia parametreja, jotka on syytä säätää ratkaisun tarpeiden mukaisiksi. Aina ei esimerkiksi ole tarpeellista käyttää laskentateholtaan raskainta vaihtoehtoa. [16][10]

4 Yhteenveto

Tämän tutkielman tavoitteena oli tarkastella ICT-alan energiankulutusta yleisellä tasolla ja tarkemmin sitä, miten itse ohjelmakoodi vaikuttaa energiankulutukseen. Näihin tutustumisen jälkeen pyrkimyksenä oli tutkia erilaisia tapoja, joilla ohjelmoija voi konkreettisesti vaikuttaa työnsä energiankulutukseen. Näitä tapoja löytyi aina ohjelmiston alku- ja suunnitteluvaiheessa sovellettavista käytänteistä itse ohjelmakoodin kirjoittamisvaiheessa hyödynnettäviin käytänteisiin.

Luvussa 2 käsiteltiin ICT-alan ja ohjelmistojen energiankulutusta ja siinä todettiin alan energiankulutuksen kasvavan nopeasti, joten ratkaisut tämän vähentämiseksi eivät ole epäolennaisia. Ohjelmakoodin vaikutuksen energiankulutukseen voidaan pohjimmiltaan sanoa aiheutuvan tiedonkäsittelystä ja siirrosta. Tämä vaatii prosessorilta kellosyklejä, mikä kuluttaa energiaa. Vaikka periaatteessa laitteen ja sen ohjelmistojen energiankulutusta voidaan mitata sen kuluttaman sähkötehon määrästä, saadaan tällä mittaustavalla vain laitteen yleinen energiankulutus selville, eikä päästä pureutumaan tarkemmin eri komponenttien ja prosessien energiankulutukseen, mikä on huomattavasti vaikeampi ja epäselvempi mittauskohde.

ICT-alan trendit ovat yksi merkittävä tekijä alan energiankulutuksen kasvussa. Trendit ovat vieneet kehitystä suuntaan, jossa ohjelmistojen ominaisuudet ovat tärkeämpiä, kuin niiden tehokkuus. Itse ohjelmointi puolestaan tapahtuu usein nykyisin korkeammalla abstraktion tasolla, tehden ohjelmoinnista nopeampaa ja helpompaa energiankulutuksen kustannuksella.

Tutkielman toisen tutkimuskysymyksen tarkastelu aloitettiin tutustumalla siihen, mikä tekee ohjelmistoista tehottomia. Tätä tehottomuutta lähestyttiin hukkien kautta, eli ylimääräisten ja turhien toimintojen kautta. Luvussa käsitellyt hukat aiheutuvat usein kehittäjien puutteellisesta ymmärryksestä ja siitä, että energiankulutusta ei ole huomioitu ohjelmiston suunnitteluvaiheessa. Myös ohjelmiston käyttökokemuksen puutteellinen huomiointi joko vahingossa tai tarkoituksella pimeillä käytännöillä aiheuttaa hukkaa.

Ohjelmistojen tehottomuuteen tutustumisen jälkeen tutkittiin keinoja, joilla ohjelmoija voi välttää tätä tehottomuutta. Nämä vihreät menetelmät voidaan karkeasti tiivistää hyvään ja perusteelliseen ohjelmiston suunnitteluun ja käytettävien teknologioiden ja työkalujen ymmärtämiseen. Jos ohjelmiston käyttötarkoitus ja näin

ollen sen vaatimat ominaisuudet ovat hyvin selvillä, voidaan välttyä turhilta ominaisuuksilta ja niiden aiheuttamilta hukilta. Vastaavasti ohjelmoijalla pitää olla riittävä ymmärrys käyttämistään teknologioista ja työkaluista, jotta hän osaa ensinnäkin valita oikean työkalun oikeaan tehtävään ja myös käyttää sitä oikein ja näin välttyä puutteellisen tiedon ja taidon aiheuttamilta hukilta.

Tulevaisuudessa olisi kannattavaa panostaa yksittäisten ohjelmistojen ja prosessien energiankulutuksen mittaamiseen, koska sen todettiin tutkielman havaintojen mukaan olevan haasteellista. Tätä mittaamista voitaisiin tehdä ohjelmistojen elinkaareen aikana esimerkiksi vertailemalla ohjelmiston eri versioiden energiankulutusta. Näin saataisiin parempi kuva tehtyjen muutosten vaikutuksesta energiankulutukseen ja kehitystä pystyttäisiin ohjaamaan saatujen mittaustulosten mukaisesti. Vastaavasti ohjelmistojen tarjoajat voisivat tuoda mittaustuloksiaan käyttäjien tietoon, jolloin energiatehokkaiden ohjelmistojen valitseminen olisi helpompaa. Energiatehokkuuden läpinäkyvyyden lisääminen kasvattaisi myös sen kilpailuvalttia, jolloin alan toimijoilla olisi suurempi kiinnostus käyttää vihreän ohjelmoinnin menetelmiä kehitysprosesseissaan.

Tutkielman tutkimuskysymyksiin vastattiin onnistuneesti. ICT-alan ja itse ohjelmakoodin energiankulutuksen kasvun voidaan todeta olevan ongelma, jonka ratkaisemiseen on olemassa vihreän ohjelmoinnin menetelmiä. Näitä menetelmiä hyödyntämällä, ohjelmoijan on mahdollista vaikuttaa oman työnsä energiankulutukseen. Tämä on hyödyllistä paitsi energiankulutuksen kannalta, myös ohjelmoijan tietotaito kasvaa, koska monet ohjelmistojen tehottomuutta aiheuttavat tekijät ovat yksinkertaisesti virheitä ohjelmoijan osalta ja näiden korjaaminen tekee hänestä paremman työssään.

5 Lähteet

[1] Radersma, Reinder. "Green Coding: Reduce Your Carbon Footprint." Ethical Software Engineering (2022): 19

[2] Ross, A., & Christie, L. (n.d.). Energy Consumption of ICT. UK Parliament Post

[3] Ward Van Heddeghem, Sofie Lambert, Bart Lannoo, Didier Colle, Mario Pickavet, Piet Demeester, Trends in worldwide ICT electricity consumption from 2007 to 2012, Computer Communications, Volume 50, 2014, Pages 64-76, ISSN 0140-3664

[4] https://www.ictfootprint.eu/

[5] E. Jagroep *et al.*, "Awakening Awareness on Energy Consumption in Software Engineering," *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Society Track (ICSE-SEIS)*, Bueons Aires, Argentina, 2017, pp. 76-85, doi: 10.1109/ICSE-SEIS.2017.10

[6] Jukka Manner, Black software — the energy unsustainability of software systems in the 21st century, *Oxford Open Energy*, Volume 2, 2023, oiac011, https://doi.org/10.1093/ooenergy/oiac011

[7] Pereira R, Couto M, Ribeiro F et al. Energy Efficiency across Programming Languages. In: Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering, October 2017, 2017, 256–6

[8] https://pypl.github.io/PYPL.html

[9] Wirth, Niklaus. "A plea for lean software." Computer 28.2 (1995): 64-68

[10] Kalliola, Janne. “Vihreä koodi” 2023-08-31

[11] M. Pärssinen, M. Kotila, R. Cuevas, A. Phansalkar, J. Manner, Environmental impact assessment of online advertising, Environmental Impact Assessment Review, Volume 73, 2018, Pages 177-200, ISSN 0195-9255

[12] Ahmad, Tanveer, and Dongdong Zhang. "A critical review of comparative global historical energy consumption and future demand: The story told so far." Energy Reports 6 (2020): 1973-1991

[13] Yokoyama, André M., et al. "Investigating hardware and software aspects in the energy consumption of machine learning: A green AI‐centric analysis." Concurrency and Computation: Practice and Experience 35.24 (2023): e7825

[14] Wells, John D., et al. "The effect of perceived novelty on the adoption of information technology innovations: a risk/reward perspective." Decision Sciences 41.4 (2010): 813-843

[15] Mehrotra, Dheeraj. Basics of artificial intelligence & machine learning. Notion Press, 2019

[16] Markelius, Alva, et al. "The mechanisms of AI hype and its planetary and social costs." AI and Ethics (2024): 1-16