Jenni Yrjänä

Kehittämistutkimus: Informatiivinen kysely vihreän ohjelmointiosaamisen tukemiseksi

Tietotekniikan
pro gradu -tutkielma
14. toukokuuta 2024

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Kokkolan yliopistokeskus Chydenius

Tekijä: Jenni Yrjänä
Yhteystiedot: jenni.a.yrjana@jyu.fi
Puhelinnumero: 040-864 9772
Ohjaaja: Lasse Harjumaa
Työn nimi: Kehittämistutkimus: Informatiivinen kysely vihreän ohjelmointiosaamisen tukemiseksi
Title in English: Design Science Research: An informative survey to support green software development
Työ: Tietotekniikan pro gradu -tutkielma
Sivumäärä: 111+12
Tiivistelmä: Vihreästä ohjelmoinnista on tullut tärkeä osa ohjelmointiosaamista. On kuitenkin tärkeää, että ohjelmistokehittäjät tuntevat vihreän ohjelmoinnin käytänteitä ja osaavat hyödyntää niitä ohjelmistotuotannossa. Tässä työssä tehtiin tutkittuun tietoon perustuva vihreää ohjelmistokehitysprosessia ja ohjelmointia koskeva informatiivinen sähköinen kysely. Tutkielma toteutettiin yhteistyössä suomalaisen ICT-ratkaisuja ja -palveluja eri toimialoille toimittavalle ohjelmistoyhtiö Digia Oyj:n kanssa. Tutkimusmenetelmänä käytettiin Design Science Research Methodology -prosessia. Tutkimuksen empiirisessä osassa hyödynnettiin teemahaastattelua.

Tutkimuksen tuloksena syntyneellä artefaktilla pyritään edistämään ohjelmistokehitykseen osallistuvien työntekijöiden vihreää ohjelmistokehitysprosessia ja ohjelmointia koskevaa tietämystä. Luotu artefakti edistää työntekijöiden tietotaitoja vihreästä ohjelmoinnista kysymysten ja välittömän informatiivisen palautteen avulla. Kyselyn lopussa annetaan palaute kyselyn tuloksista sekä ohjataan kyselyyn vastaaja hakemaan tarvittaessa lisäinformaatiota aiheeseen liittyviltä nettisivuilta. Haastateltavat pitivät kyselyä tehokkaana ja hyvänä tapana lisätä ohjelmistokehittäjien tietämystä vihreistä ohjelmointikäytänteistä. Kyselyllä pystyttiin lisäämään ohjelmistokehittäjien tietämystä vihreistä ohjelmointikäytänteistä.
Avainsanat: Vihreä IT, vihreä ohjelmistokehitys, kehittämistutkimus, informatiivinen kysely
Abstract: Green programming has become an important part of programming skills. However, it is important that software developers know the practices of green programming and know how to use them in software production. In this work, an informative electronic survey about the green software development process and programming based on researched information was made. The study was carried out in cooperation with the Finnish software company Digia Oyj, which supplies ICT

solutions and services to various industries. The Design Science Research Methodology process was used as a research method. Furthermore, in the empirical part of the study, a theme interview was utilized.

The artifact, which was created as a result of the research, aims to promote the green software development process and programming knowledge of the employees involved in software development. The created artifact promotes employees knowledge of green programming through questions and immediate informative feedback. At the end of the survey, feedback is given on the results of the survey and the survey respondent is directed to search for additional information on the related websites if necessary. The interviewees considered the survey to be an effective and good way to increase software developers knowledge of green programming practices. The survey was able to increase software developers knowledge of green programming practices.
Keywords: Green IT, Green software development, Design Science Research, Informative survey

Sanasto

Black-box testing	Musta laatikko -testaus
Bulk Operation	Massaoperaatio
DRAM	Dynamic Random Access Memory
FaaS	Function-as-a-Service
FASTA	DNA and protein sequence alignment software package
I/O	Input/Output
IoT	Internet of Things, Esineiden Internet
Lazy Initialization	Laiska initialisointi
Peak Memory	Huippumuisti
PUE	Power Usage Effectiveness, Virrankäytön tehokkuus
TDD	Test-Driven Developing, Testilähtöinen kehittäminen
TDRE	Test-driven requirements engineering, Testilähtöinen vaatimusmäärittely
White-box testing	Valkoinen laatikko -testaus

Sisällys

Liitteet

1 Johdanto

Viime vuosikymmeninä tietotekniikan määrä on kasvanut huomattavasti. Sitä mukaa, kun laitteiden ja ohjelmistojen määrä kasvaa, myös niiden tarvitsema energia kasvaa huomattavasti. [61] Mitä enemmän tietotekniikkaa ja ohjelmistoja käytetään, sitä suuremman hiilijalanjäljen ne jättävät. Jokainen koodirivi, joka kirjoitetaan tänään, voi olla käynnissä vielä vuosien päästä valtavassa määrässä prosessoreita. Sen takia ohjelmistokehittäjien tulisi kiinnittää erityistä huomiota hyviin vihreisiin käytänteisiin, joilla voidaan pienentää ohjelmistokehitysprosessin sekä itse ohjelmistoon liittyvää hiilijalanjälkeä. Vihreän IT:n tarkoituksena on erityisesti vaikuttaa ilmastonmuutoksen hillitsemiseen, mutta tämän tutkielman kirjoittamisen ajankohdalle sattuvan energiakriisin aikana, aihe on vielä enemmän käsinkosketeltava kuin aikaisemmin.

Ohjelmistokehittäjille on tarjolla paljon yksityiskohtaisia ohjelmistotekniikoita vihreän ohjelmiston rakentamiseksi. Näitä tekniikoita hyödyntämällä voidaan pienentää ohjelmistokehitysprosessin ja ohjelmiston energiankulutusta ja hiilidioksidipäästöjä. Näihin tavoitteisiin pääseminen kuitenkin vaatii ohjelmistokehittäjien olevan tietoisia ohjelmistokehitysprosessin ja ohjelmiston energiankulutukseen ja hiilidioksidipäästöihin vaikuttavista asioista. Heillä täytyy olla myös osaamista tehdä näistä vähemmän energiaa kuluttavia, vähemmän hiilidioksidipäästöjä tuottavia ja siten vihreämpiä. Työnantajilla onkin haasteena varmistaa työntekijöidensä vihreä ohjelmointiosaaminen.

Kuluttajat ja sijoittajat haluavat tuotteiden sekä yritysten olevan ekologisesti yhä kestävämpiä. Hiilijalanjäljen pienentämiseksi on kehitetty uusiutuvia energianlähteitä, mutta se ei ratkaise varsinaista ongelmaa, joka on liiallinen energian käyttäminen. Siksi tarvitaan vihreitä ohjelmistoratkaisuja, jotka vähentävät energiankulutusta itsessään. Yrityksille tämä tarkoittaa sitä, että toimintatapojen ja tuotteiden tulee olla ekologisia ja kestävän kehityksen mukaisia. Vihreiden arvojen näkymisen koetaan vaikuttavan positiivisesti yritykseen niin työyhteisön sisällä kuin yrityksen kilpailukykyynkin [11]. Ahmad et al. [6] tutkimustuloksista käy myös ilmi, että kaikki yritykset eivät ole adaptoineet vihreitä käytänteitä ohjelmistokehitysprosessiin. Ohjelmistokehittäjät välittävät ohjelmiston energiankulutuksesta, mutta ei-

vät onnistu sen vähentämisessä parhaalla mahdollisella tavalla, koska heidän tietämyksensä aiheesta on puutteellinen. He myös uskovat voivansa oppia parantamaan ohjelmiston energiatehokkuutta ohjelmistokehityksen eri vaiheissa. [36] Ohjelmiston vihreys täytyy ottaa huomioon kaikissa ohjelmistokehitykseen liittyvissä vaiheissa. Energiatehokkuus voidaan ajatella koskevan jokaista ohjelmistokehitysprojektin vaihetta aina suunnittelusta testaukseen, koodaukseen, käyttöönottoon, käyttöön sekä käytöstä luopumiseen. Ohjelmistosta voidaan tehdä vihreämpi pitämällä itse kehitysprojekti vihreänä, mutta myös huolehtimalla itse tuotteen kestävyydestä. Kehitysprosessin monimutkaisuuden takia kehittäjät tarvitsevat sekä selkeitä vihreitä ohjelmistokehitysprosessin malleja että käytännön ohjeita siitä miten ympäristöystävällisen ohjelmiston rakentaminen on mahdollista.

Tutkimuksen päätavoitteena on kehittää informatiivinen kysely, jolla voidaan arvioida ja lisätä ohjelmistokehittäjien tietämystä vihreän ohjelmistokehitysprosessin ja vihreän ohjelmiston käytänteiden suhteen. Tavoitteena on myös lisätä tietämystä vihreästä ohjelmoinnista ja vaikuttaa positiivisesti ohjelmistokehittäjien suhtautumiseen vihreää ohjelmistokehitystä kohtaan. Samalla tavoitteena on kartoittaa tutkimukseen osallistuvien mielipiteitä siitä, onko kysely tehokas metodi kartoittaa ja lisätä ohjelmistokehittäjien vihreää ohjelmointiosaamista ja motivoiko kysely implementoimaan vihreitä ohjelmointikäytänteitä käytännön työhön. Tutkimus toteutetaan kehittämistutkimuksena Digia Oyj:n asiakasprojekteja toteuttavassa organisaatioyksikössä.

Tutkimuksen tutkimuskysymykset ovat seuraavat:

1. Mitkä ovat keskeisimmät vihreän ohjelmistokehityksen käytänteet?
2. Millaisella informatiivisella kyselyllä voidaan edistää vihreiden ohjelmistokehitykseen liittyvien käytänteiden tietämystä?
3. Miten vihreän ohjelmoinnin käytänteet toteutuvat organisaatiossa?

Kehittämistutkimuksen empiirinen osa toteutetaan teemahaastattelujen avulla ja näitä tuloksia hyödynnetään kehitettäessä informatiivista kyselyä. Tuloksena kehitetään Digia Oyj:lle iteratiivisesti sähköinen informatiivinen kysely. Tämän kyselyn aiheet ovat valikoituneet kirjallisuuskatsauksen perusteella, johon on kerätty keskeisimpiä vihreään ohjelmointiin liittyviä käytänteitä. Haastattelujen tuloksissa selvitetään myös vihreiden käytänteiden toteutumista organisaatiossa.

Luvussa kaksi kerrotaan tutkimuksen toteuttamisesta. Luvussa kolme kerrotaan kestävän kehityksen mukaisesta ohjelmistosta ohjelmistokehitysprosessin, ohjelmis-

ton elinkaarimallin ja ohjelmoinnissa huomioitavien käytänteiden näkökulmasta. Luvussa neljä kerrotaan lyhyesti kyselyn rakentamisesta. Luvussa viisi käydään ensimmäisen iteraation kulku, teemahaastattelun tulokset sekä tuotos. Luvussa kuusi käydään läpi toisen iteraation kulku, teemahaastattelun tulokset sekä tuotos. Luvussa seitsemän käydään läpi tutkimuksen tulokset, luvussa kahdeksan pohdinta ja luvussa yhdeksän yhteenveto.

2 Tutkimuksen toteuttaminen

Tutkimus toteutettiin laadullisena kehittämistutkimuksena, jonka aikana kehittämistuotoksena syntyi informatiivinen sähköinen kysely vihreästä ohjelmistokehitysprosessista ja ohjelmointikäytänteistä. Tutkimus toteutettiin yhteistyössä Digia Oyj:n kanssa ja kysely on tarkoitettu ohjelmistoprojektien kanssa työskenteleville edistämään vihreän ohjelmistokehityksen käytännön osaamista.

Tutkimuksen ensimmäisessä vaiheessa tutustuttiin kirjallisuuden avulla vihreän ohjelmistokehitysprosessin ja ohjelmiston periaatteisiin ja näiden tietojen avulla rakennettiin alustava kysely. Kyselyn arviointiin osallistui Digia Oyj:n Green Code Expert ja toisessa ja kolmannessa vaiheessa ohjelmistokehityksen parissa työskenteleviä henkilöitä Digia Oyj:ltä. He osallistuivat kyselyn testaamiseen sekä heiltä kerättiin teemahaastattelujen avulla aineistoa kyselyn kehittämiseksi. Seuraavissa alaluvuissa esitellään tutkimuksen tutkimusmenetelmä ja sen valintaan johtaneet perustelut, kehittämistutkimuksesta ja sen soveltamisesta tässä tutkimuksessa sekä teemahaastattelusta aineistonkeruumenetelmänä, aineiston analysoimisesta sekä pohdintaa tutkimuksen luotettavuudesta.

2.1 Tutkimuksen tarkoitus, tavoitteet ja tutkimuskysymykset

Manotas et al. [36] ovat tutkineet ohjelmistokehittäjien näkemyksiä vihreästä ohjelmoinnista ohjelmistokehitysprosessin eri vaiheissa. He havaitsivat, että ohjelmistokehittäjät uskovat voivansa oppia parantamaan energiatehokkuutta. Tämä viittaa siihen, että heiltä puuttuu tarvittava tieto ja asiantuntemus voidakseen tehdä näin. Tutkijat havaitsivat myös, että ohjelmistokehittäjillä on halu oppia tuottamaan vihreitä ohjelmistoja ja he olisivat valmiita käyttämään esimerkiksi muiden kehittäjien ammattitaitoa, työkaluja, profilointia, esimerkkikoodeja ja dokumentaatioita tämän saavuttamiseksi. On todennäköistä, että ohjelmistokehittäjät ottaisivat koulutuksen vastaan hyvin sen kaikissa muodoissa.

Tämän tutkimuksen tarkoituksena on kehittää teoriaan ja organisaation toimintatapoihin pohjautuva informatiivinen kysely vihreään ohjelmointiin liittyvän tietämyksen arvioimiseksi sekä kartoittaa tutkimukseen osallistuvien kokemuksia kyse-

lystä.

Tutkimuksen tavoitteet ovat:

1. Kehittää informatiivinen kysely, jolla voidaan arvioida ohjelmistokehittäjien tietämystä vihreän ohjelmistokehitysprosessin ja vihreän ohjelmiston tekijöiden suhteen.
2. Lisätä tietoa vihreästä ohjelmoinnista ja vaikuttaa positiivisesti ohjelmistokehittäjien suhtautumiseen vihreää ohjelmistokehitystä kohtaan.
3. Kartoittaa tutkimukseen osallistuvien mielipiteitä siitä, onko kysely tehokas metodi kartoittaa ja lisätä ohjelmistokehittäjien vihreää ohjelmointiosaamista.
4. Selvittää motivoiko kysely implementoimaan vihreitä ohjelmointikäytänteitä käytännön työhön.

Tutkimuskysymykset ovat:

1. Mitkä ovat keskeisimmät vihreän ohjelmistokehityksen käytänteet?
2. Millaisella informatiivisella kyselyllä voidaan edistää vihreiden ohjelmistokehitykseen liittyvien käytänteiden tietämystä?
3. Miten vihreän ohjelmoinnin käytänteet toteutuvat organisaatiossa?

2.2 Tutkimusmenetelmän valinta

Tämä tutkielma toteutettiin kehittämistutkimuksena, jossa aineistonkeruu toteutettiin puolistrukturoidun teemahaastattelun avulla. Tarkoituksena oli soveltaa toimintatutkimuksen prosessikaaviota sekä huomioida, että laadukkaan suunnittelutieteellisen tutkimuksen kriteerit täyttyvät. Tutkimusote valikoitui Digian tarpeesta tehdä vihreään ohjelmistokehitykseen liittyviä parannuksia ja selvittää ohjelmistokehittäjien tietämystä vihreästä ohjelmistokehityksestä. Kehittämistutkimus sopi tällaiseen asetelmaan hyvin.

2.3 Kehittämistutkimus tutkimusmenetelmänä

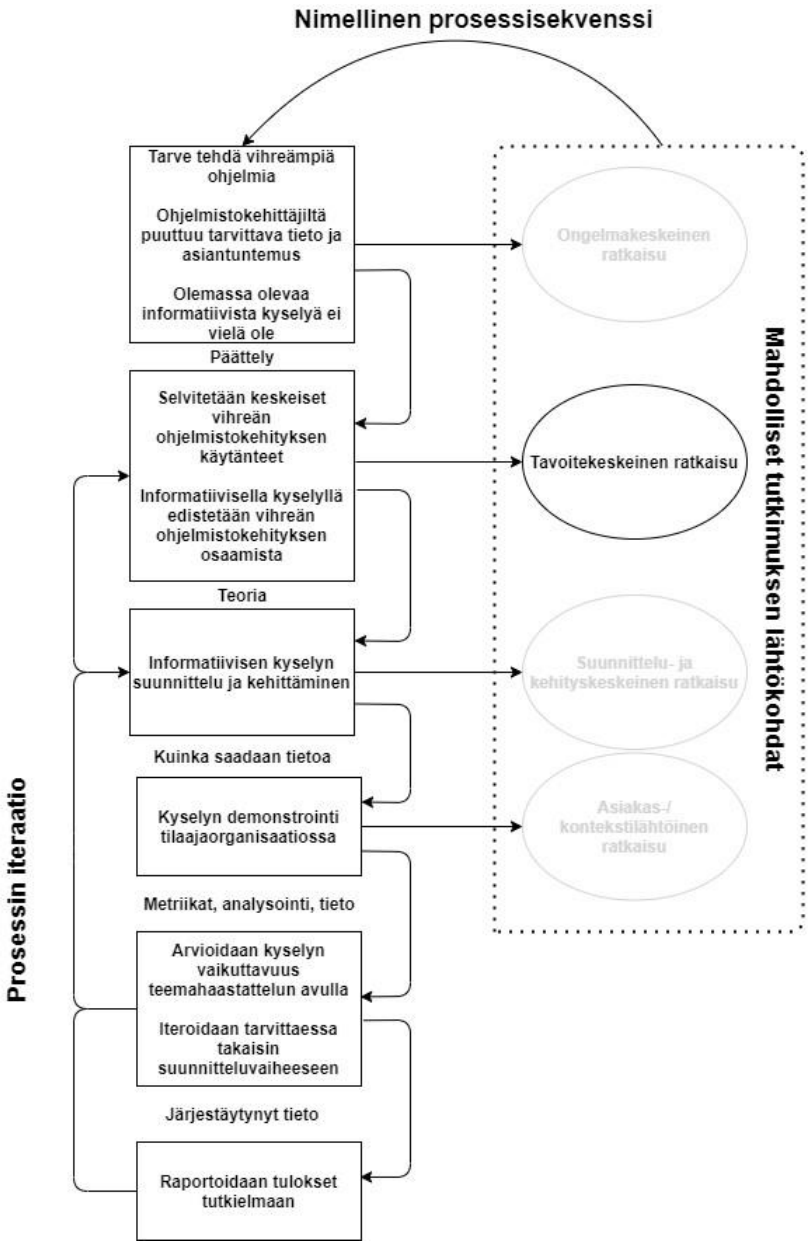
Kehittämistutkimus ei ole erillinen tutkimusmenetelmänsä, vaan erilaisista tutkimusmenetelmistä rakennettu kokonaisuus. Erilaisia tutkimusmenetelmiä sovelletaan kehittämiskohteen tarpeiden mukaisesti. Kyseessä on siis monimenetelmäinen tutkimusstrategia. Kehittämistutkimuksen tarkoituksena on kehittää jotain asiaa paremmaksi tai poistaa ongelma. Kehittämistutkimus jakaantuu kahteen erilliseen prosessiin, joista toinen koskee itse tutkimusta ja toinen kehittämistyötä. Kehittämistyön prosessi noudattaa sille tyypillistä kehittämisprosessia ja tutkimustyö noudattaa tutkimusmenetelmän mukaista tutkimusprosessia. [26, s. 19]

Kehittämistutkimus perustuu aina teoriaan, johon kehittäminen perustetaan. Sen lisäksi kehittämistutkimus vaatii tutkimuksellisen osion, jotta voidaan puhua tutkimuksesta. Kehittämistutkimuksen kohteena voi olla prosessi, toiminto, asiantila tai tuote. [26, s. 19] Hevner et al. [21] on määritellyt hyvän suunnittelutieteellisen tutkimuksen tekemiseen ja arviointiin tutkimusohjeet, jotka tutkijan tulisi ottaa huomioon kehittämistutkimusta toteuttaessaan. Ensimmäinen kriteeri on, että kehittämistutkimuksen tulee tuottaa käytettävissä oleva artefakti konstruktion, mallin, menetelmän tai ilmentymän muodossa. Toinen kriteeri on se, että artefaktin tulee olla luotu ratkaisemaan jokin tietty ongelma. Kolmas kriteeri on se, että ehdotetun artefaktin hyödyllisyys ja laatu on osoitettava ja arvioitava hyvin toteutettujen arviointimenetelmien avulla. Neljäs kriteeri on, että tutkimuksen vaikuttavuus tulee selkeyttää organisaatiolle sekä akateemiselle yleisölle, jotta kiinnostus ja tietämys aihetta kohtaan kasvaisi. Viides kriteeri on, että tutkimus suoritetaan riittävän tarkasti ja kurinalaisesti, jotta voidaan varmistua artefaktin soveltuvuudesta sen käyttötarkoitukseen, että se täyttää kehittämiskriteerit ja on luotettava. Kuudes kriteeri on se, että tutkijan täytyy tehdä tutkimusta ymmärtääkseen tutkimusongelmaa ja saadakseen siihen ongelmanratkaisumenetelmiä. Seitsemäs kriteeri on, että tutkimuksen tuloksista tulee raportoida asianmukaisesti kaikille asianomaisille.

Suunnittelutieteellisiä tutkimusprosessin malleja on useita [14] [26, s. 53]. Osa prosessimalleista kuvataan vesiputousmallina ja osa prosessimalleista sisältää silmukoita, joita toistetaan, kunnes artefakti on tyydyttävällä tasolla. Kaikki prosessimallit eroavat toisistaan yksityiskohdissa, mutta ovat pääpiirteittäin samanlaisia. Prosessimalleja tarkastellessa on huomioitavaa myös se, että kaikki mallit eivät vastaa Hevner et al. [21] asettamia kriteerejä laadukkaalle suunnittelutieteelliselle tutkimukselle. Peffers et al. [44] ovat esittäneet tutkimuksessaan prosessikaavion, jota on sovellettu tämän tutkimuksen läpiviemiseksi. 2.1. Tämä kaavio koostuu kuudesta

eri prosessin vaiheesta. Ensimmäisessä vaiheessa identifioidaan tutkimusongelma ja tutkimuksen motivaatio, määritellään ongelma tarkemmin ja perustellaan, miksi tämän tutkimuksen tekeminen on tärkeää. Toisessa vaiheessa määritellään ratkaisun tavoitteet ja kerrotaan mitä artefaktilla voitaisiin saavuttaa. Kolmannessa vaiheessa suunnitellaan ja kehitetään artefakti. Neljännessä vaiheessa demonstroidaan artefaktin käyttöä. Viidennessä vaiheessa arvioidaan artefaktin vaikuttavuutta ja tehokkuutta. Tässä vaiheessa voidaan palata takaisin toiseen tai kolmanteen vaiheeseen eli palataan toistamaan iteraatioprosessi. Viimeisenä vaiheena kaaviossa julkistetaan tulokset esimerkiksi tieteelliseksi julkaisuksi tai ammattijulkaisuksi. Kaaviossa esitetään myös mahdolliset tutkimukselliset lähtökohdat, joita ovat ongelmakeskeinen, tavoitekeskeinen, suunnittelu ja kehityskeskeinen sekä asiakas-/ kontekstikeskeinen ratkaisu.

Tässä tutkimuksessa on pyritty noudattamaan kyseessä olevaa prosessimallia. Ongelman lähtökohdat saatiin Digialta Oyj:lta, jonka tavoitteena on tehdä tulevaisuudessa vihreämpiä ohjelmistoja. Ongelmana on kuitenkin se, että ohjelmistokehittäjiltä puuttuu tarvittava tieto ja asiantuntemus voidakseen vastata alkuperäiseen tavoitteeseen. Digia Oyj:lla on käytössään koulutustarkoitukseen esimerkiksi tietoturvaan liittyviä kyselyitä, mutta vihreään ohjelmistokehitykseen liittyvää informatiivista kyselyä ei vielä ollut. Tavoitteeksi asetettiin kehittää vihreään ohjelmointiin liittyvä informatiivinen kysely ja prosessin seuraavana vaiheena oli selvittää keskeiset vihreän ohjelmistokehityksen käytänteet. Tultiin siihen tulokseen, että informatiivisella kyselyllä voitaisiin edistää vihreän ohjelmistokehityksen osaamista. Hankittuun teoriapohjaan perustuen suunniteltiin ja kehitettiin ensimmäinen versio kyselystä. Kyselyä demonstroitiin ensiksi Digia Oyj:n yhteyshenkilölle ja tämän jälkeen heidän kuudelle työntekijälleen. Vaikuttavuutta arvioitiin teemahaastattelun avulla ja tämän jälkeen palattiin takaisin suunnittelemaan ja kehittämään kyselyä edelleen. Prosessissa toistettiin kyselyn demonstrointi ja arvioitiin kyselyn toisen version vaikuttavuus haastattelemalla neljää toimeksiantajan työntekijää. Prosessissa palattiin takaisin kyselyn kehittämiseen haastattelun tulosten perusteella. Kyselyn kolmas versio demonstroitiin toimeksiantajan yhteyshenkilölle, joka hyväksyi valmistuneen artefaktin. Tässä vaiheessa poikettiin prosessikaaviosta ja teemahaastattelun avulla tapahtuvaa arviointia enää suoritettu. Viimeisenä vaiheena toteutettiin tämän tutkielman kirjoittaminen. Kehittämistutkimuksen aikana toteutettiin prosessin iteraatio kokonaisena yhteensä kaksi kertaa.



Kuva 2.1: Kehittämistutkimuksen kehittämissyklin vaiheet. Mukaillen [44]

2.4 Aineistonkeruu teemahaastattelulla

Tässä tutkielmassa aineisto kerättiin teemahaastattelun avulla. Teemahaastattelu on puolistrukturoitu haastattelu, jossa käytetään haastattelurunkoa tarkkojen kysymysten sijaan. Haastattelurunko koostuu teemoista ja tarkkoja kysymysmuotoja ei määritellä. Haastattelu voi myös edetä joustavasti ja sen ei tarvitse noudattaa ennalta määrättyä järjestystä. Teemojen ansiosta tiedonantajat voivat kertoa aiheesta omin sanoin ja he voivat tuoda esille myös sellaisia näkökulmia, joita tutkielman kirjoittaja ei ehkä ole osannut ajatella. Tällä tavalla voidaan myös vähentää tutkijan omien ennakko-odotusten vaikutusta tutkimustuloksiin. Vihreä ohjelmointi on laaja aihealue, joten on odotettavaa, että tutkimuksen aihe tuottaa vastauksia tutkimuksen kohteena olevan kyselylomakkeen aihealueiden ulkopuolelta. Teemahaastattelun aikana tutkijalla on mahdollisuus syventää saatuja tietoja tai selventää saatuja vastauksia. Näiden asioiden takia teemahaastattelun valinta on perusteltua. [22, s. 47–49] [23, s. 204–205, 208]

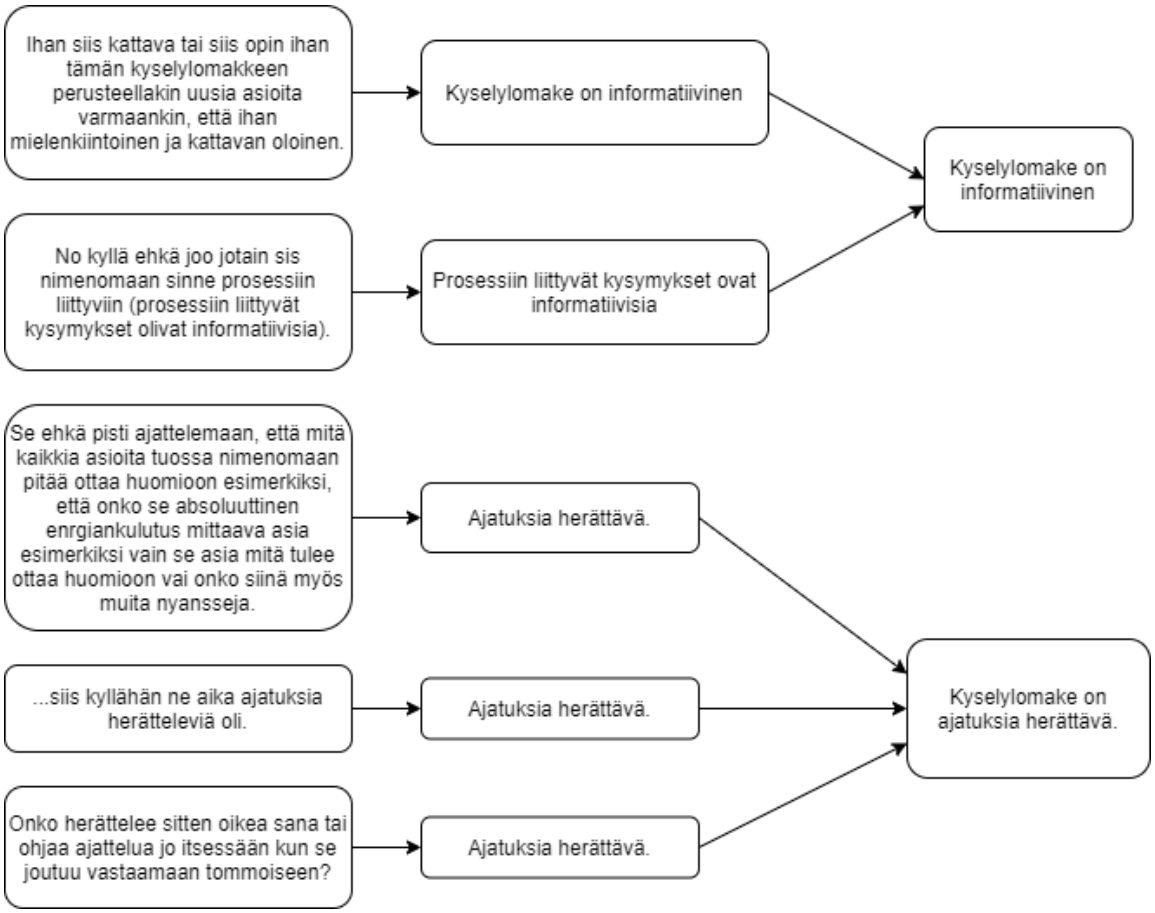
Haastattelut toteutettiin Microsoft Teams -sovelluksen avulla. Tällä ohjelmistolla voidaan sopia ja toteuttaa etätapaamisia ja tapaamiset voidaan myös nauhoittaa mp4-tallenteiksi. Nauhoitteet tallentuivat automaattisesti Jyväskylän yliopiston pilvipalvelimelle. Haastattelujen apuna käytettiin haastattelurunkoja 5.2 ja 6.2. Haastateltavien työkokemus ja asema työyhteisössä olivat erilaisia ja haastatteluja pyrittiin syventämään haastateltavien erikoisosaamisen perusteella. Ensimmäiset haastattelut olivat kestoltaan noin 45–75 minuuttia. Seuraavat haastattelut olivat lyhyempiä, noin 10–25 minuuttia.

Aineiston analysointi suoritettiin induktiivisella sisällönanalyysimenetelmällä, joka on perinteinen laadullisen tutkimuksen analysointimenetelmä. Sisällönanalyysin vahvuutena on se, että sillä voidaan tavoittaa merkityksiä, seurauksia ja sisältöjä sekä kuvata ilmiöiden välisiä suhteita. Sisällönanalyysi haastaa tutkijan prosessoimaan tuloksia aktiivisesti. [58, s. 91]

Aineisto litteroitiin Microsoft Word -tekstinkäsittelyohjelman litterointityökalua hyödyntäen ja tämän jälkeen puhtaaksikirjoitettiin poistamalla täytesanoja, toistoja sekä tutkimukselle merkityksettömiä lauseita. Litteroimisen jälkeen aineistoon tutustuttiin syvällisemmän kokonaiskuvan saamiseksi. Seuraavana vaiheena päätettiin aineistossa olevat kiinnostavat asiat, kerättiin yhteen ja siirrettiin erilleen muusta aineistosta jatkokäsittelyä varten. Viimeisenä vaiheena aineisto teemoitettiin, jolloin aineisto pilkottiin ja ryhmiteltiin aihepiirin mukaisesti. [58, s. 92–93]

Sisällönanalyysi sopi tähän tutkimukseen, koska tarkoituksena oli kuvata aineis-

ton sisältöä sanallisesti sekä saamaan ilmiöstä kuvaus tiivistetyssä muodossa. Sisällönanalyysillä pystyttiin järjestämään saatu aineisto tiiviiseen ja selkeään muotoon. Näin aineistosta pystyttiin luomaan selkeää ja yhtenäistä informaatiota, jotta tutkittavasta ilmiöstä voitiin tehdä selkeitä ja luotettavia johtopäätöksiä. [58, s. 103] Kuvassa 2.2 esitetään esimerkki suoritetusta sisällönanalyysistä. Sisällönanalyysi on saatavissa pyydettäessä kokonaisuudessaan.



Kuva 2.2: Esimerkki sisällönanalyysistä.

2.5 Tutkimuksen luotettavuus

Koska kehittämistutkimus ei ole oma tutkimusmenetelmänsä, luotettavuutta tulee arvioida niiden menetelmien luotettavuuskriteereillä, joita on käytetty tutkimusta tehtäessä. Perinteisesti tieteellisten tutkimusten luotettavuutta on arvioitu validiteetin sekä reliabiliteetin avulla. Nämä käsitteet ovat kuitenkin peritty määrällisen

tutkimuksen puolelta, ja niitä ei voi sellaisenaan soveltaa laadulliseen tutkimukseen. Tavallinen tapa tarkastella laadullisen tutkimuksen luotettavuutta on soveltaa Lincolnin ja Guban kehittämää luokittelua [33]. Tämä luokittelu koostuu neljästä luokasta: uskottavuus, siirrettävyys, luotettavuus sekä vahvistettavuus. Näistä luokitteluista on useita erilaisia suomennoksia sekä tulkintoja. [58, s. 138–139] [23, s. 231–233] [26, s. 172]

Kehittämistutkimuksen luotettavuutta tulisi kuitenkin arvioida myös kehittämistutkimuksen ominaispiirteiden näkökulmasta. Kehittämistutkimuksen luonteeseen kuuluu vaiheittainen tuotekehitys, joten on perusteltua arvioida tutkimuksen luotettavuutta prosessivaliditeetin näkökulmasta. Prosessivaliditeettia arvioidessa otetaan huomioon erityisesti tutkimusprosessin hallinta ja johdonmukaisuus. Pohdittavia asioita ovat esimerkiksi se, onko tutkimusta teoreettisia ja käytännöllisiä lähtökohtia kehitetty iteratiivisesti, onko kehittämistyön tuloksia hyödynnetty seuraavassa vaiheessa ja onko kehittämistutkimuksen iteratiivinen luonne huomioitu raportoinnissa. Tässä arvioidaan tutkimukseen osallistuvien yhteistyötä ja pohdittavia asioita ovat tutkimukseen osallistuneiden asiantuntijuusalueiden moninaisuus ja onko raportissa kuvattu osallisten yhteistyötä. [60, s. 243–248] Tämä tarkoittaa siis kehittämistutkimuksessa hyvää tutkimusprosessin vaiheiden ja tulosten dokumentaatiota. Dokumentaatiolla voidaan todentaa se mitä on tehty ja miten toimittu. Aineiston tuottamisen olosuhteista kerrotaan selkeästi ja totuudenmukaisesti. Tämä tarkoittaa esimerkiksi sitä, että kerrotaan haastatteluihin käytetystä ajasta, häiriötekijöistä, mahdollisista virhetulkinnoista ja tutkijan oma itsearviointi tilanteesta. [23, s. 232]

Kehittämistutkimuksen toisena tarkoituksena itse tuotteen kehittämisen lisäksi on teoreettisten näkökulmien jäsentely, joten luotettavuutta on tarkoituksenmukaista tarkastella erityisesti yleistettävyyden näkökulmasta. Yleistettävyyttä arvioidaan analyyttiseltä kannalta eli arvioidaan, tuotetaanko tutkimuksen tuotteesta tai aihepiiristä käsitteellistä tai teoreettista tietoa, jota voitaisiin soveltaa muissa yhteyksissä. Toisena näkökulmana pohditaan sitä, onko tutkielmassa kuvattu kehittämisen kohteena ollutta kontekstia riittävästi, että tuotetta voitaisiin hyödyntää muissa kehittämisympäristöissä. [60, s. 246–248]

Tämän tutkimuksen tarkoituksena on myös selvittää, voidaanko informatiivisella kyselyllä edistää tietämystä vihreän ohjelmoinnin suhteen ja motivoida ohjelmistokehittäjiä sisällyttämään asiaan liittyviä käytäntöjä käytännön työhön. Tällöin tutkimuksen luotettavuutta pitäisi arvioida myös käytännöllisen validiteetin näkökul-

masta. Tällöin tulisi arvioida sitä, onko tuote relevantti käytännöllisesti katsottuna. Tulisi myös arvioida, toimiiko tuote autenttisissa tilanteissa, saadaanko käytännössä hyödynnettäviä tuloksia ja tuotetaanko tutkimuksella kestäviä vaikutuksia. [60, s. 245–246]

Pernaa [48] kirjoittaa väitöskirjassaan, että tutkimuksen luotettavuutta voidaan arvioida vertaamalla Design-Based Research Collectiven [10] kriteereitä yhdessä Lincolnin ja Guban [33] määritelmiin tutkimuksen luotettavuudesta (suora lainaus Pernaa [48, s. 13–14] [10].):

- "*Kehittämisen tulee olla kokonaisvaltaista, jolloin kehittämistuloksena saadaan sekä ohjaavia malleja ja teorioita että kuvailevia teorioita (uskottavuus ja siirrettävyys)*"
- "*Kehittämisen tulee edetä sykleittäin ja sisältää jatkuvaa kehittämistä ja arviointia (uskottavuus, luotettavuus ja vahvistettavuus)*"
- "*Kehittämisessä tulee pyrkiä teorioihin, jotka ovat siirrettävissä kentälle opettajien tai muiden opetusalan ammattilaisten käyttöön (siirrettävyys)*"
- "*Kehittämisprosessiin tulee sisältyä testaamista autenttisissa olosuhteissa (siirrettävyys, luotettavuus ja vahvistettavuus)*"
- "*Kehittämistutkimuksen kaikki syklit tulee dokumentoida tarkasti (luotettavuus ja vahvistettavuus)*"

Kehittämistutkimuksen heikkoutena on se, että kvalitatiivisena tutkimuksena toteutettuna se toteutetaan usein pienellä otoskoolla ja se ei siten anna kattavaa kuvaa tutkittavasta perusjoukosta. Toisaalta kehittämistutkimuksen vahvuutena ajatellaan olevan tutkimustulosten yleistettävyys ja sen kyky selittää ilmiöitä sekä käytännöllisyys. Kehittämistuotokset ovat pragmaattisesta näkökulmasta katseltuna toimivia ja hyödyllisiä. [58, s. 135] [15] Luotettavan kehittämistutkimuksen edellytyksenä voidaan ajatella olevan toimivien ratkaisujen tuottaminen paikallisesti ja näiden ratkaisujen siirtäminen yleisempään käyttöön vasta tämän ensimmäisen vaiheen jälkeen. [7] Validointimetodi sekä konteksti tulisi reflektoida sidosryhmien näkökulmia ja kontekstia. [17]

2.6 Tutkimuskohde ja haastateltavat

Tutkimuksen kohdeyritys on ohjelmisto- ja palveluyritys Digia Oyj. Digia Oy perustettiin vuonna 1997 ja vuonna 2005 Digia Oy yhdistyi SysOpen Oyj:n kanssa.

Yhdistymisen jälkeen yrityksen nimeksi muodostui SysOpen Digia Oyj. SysOpen Digia Oyj:n nimi muutettiin Digia Oyj:ksi vuonna 2008 ja vuonna 2012 Digia osti Nokialta Qt-liiketoiminnan. Vuonna 2016 yrityksen toiminta jakautui siten, että Qt-liiketoiminta eriytettiin uudeksi yhtiöksi, jonka nimeksi tuli Qt Group Oyj. Tämän yhteydessä Digia uudisti strategiansa ja on jatkanut uudistuksia myös tämän jälkeen ja liikevaihto on kasvanut tämän jälkeen tasaisesti.

Digian toimialana on tarjota tietotekniikkaan, liikkeenjohtoon, atk-tarkastuksiin, sisäisiin tarkastuksiin ja riskienhallintaan liittyvää konsultointia, koulutusta ja palveluja. Digia myy alaan kuuluvaa kirjallisuutta, ohjelmistoja ja laitteita sekä harjoittaa muuta näihin liittyvää liiketoimintaa. Lisäksi Digia harjoittaa markkinointi-, myynti- ja hallintopalvelua. Se voi omistaa erilaisia arvopapereita, kiinteistöjä ja kulkuneuvoja, käydä niillä kauppaa ja vuokrata niitä. Digian omistaa osakkeenomistajat. Digian toiminnasta vastaa hallitus, valiokunnat sekä johtokunta.

Digian henkilöstön määrä vuoden 2022 joulukuun lopussa oli 1426 ja liikevaihto vuonna 2022 oli 170,8 miljoonaa euroa. Markkinastrategiassaan Digia panostaa digitalisaatioon liiketoiminnan saralla. Toimintaympäristöjen muuttuminen monimutkaisimmiksi luo tarpeen älykkään liiketoiminnan kehittämiselle sekä datan tehokkaammalle hyödyntämiselle. Automaation ja datan merkitys lisääntyy ja tekoälyn hyödyntäminen tulee olemaan yhä isommassa asemassa ja näkyy sovelluksissa sekä liiketoimintaprosesseissa. Sovellus- ja IT-järjestelmät laajenevat pistemäisistä kokonaisuuksista suuremmiksi. Digia pyrkii tarjoamaan asiakkaalle sopivia toimintamalleja ja laajoja ratkaisukokonaisuuksia erikoistuneiden palvelualueiden yksilöllisiin tarpeisiin ja huolehtii samalla ratkaisujen turvallisuudesta sekä kestävyydestä.

Digian yhteyshenkilö auttoi haastateltavien kutsumisessa haastatteluun. Haastateltavat ovat valikoituneet Digian sisäisestä projektiryhmästä. Haastateltavien työkokemus vaihteli neljästä vuodesta kahteenkymmeneenviiteen vuoteen ja heidän työnkuvansa vaihteli ohjelmistokehittäjästä projektipäällikköön sekä arkkitehdin tehtäviin saakka. Osalla haastateltavista ei ollut kokemusta ohjelmoinnista ollenkaan. Kaikille paitsi yhdelle ohjelmointiin ja ohjelmistokehitysprosessiin liittyvät käytänteet olivat tuttuja.

Haastateltavien suhtautuminen vihreyteen arvona vaihteli. Osa haastateltavista piti vihreyttä arvona tärkeänä, ja he pyrkivät henkilökohtaisessa elämässä toteuttamaan ekologisia toimintatapoja. Osa haastateltavista piti vihreyttä keskimääräisen tärkeänä tai ei juurikaan tärkeänä. Vihreyden tärkeys arvona katsottiin kuitenkin nousseen viimeisen kymmenen vuoden aikana.

Osa haastateltavista suhtautui vihreyden toteuttamiseen pragmaattisesti ja tärkeänä motivaationa vihreyden toteuttamiseen koettiin kustannussyyt. Haastateltavat katsoivat, että epäekologiset toimintatavat johtuivat usein tottumuksesta ja tiedon puutteesta. Vihreyden arvojen tärkeys maailman mittakaavassa ymmärrettiin. Osa haastateltavista oli sitä mieltä, että haluaisi lisätä vihreitä toimintatapoja omassa työssään ja vihreyden yhdistäminen työhön tuntuisi luontevalta.

Kaikki haastateltavat kokivat vihreän ohjelmointiosaamisensa huonoksi tai kokivat tarvitsevansa lisää tietoa vihreästä ohjelmointiosaamisesta.

3 Kestävän kehityksen mukainen ohjelmisto

Vihreän ohjelmistokehityksen onnistumisen kannalta ohjelmoijan tulisi tietää minkälainen hyvä vihreä ohjelmisto on sekä miten voidaan varmistua ja arvioida ohjelmiston laadukkuutta vihreästä näkökulmasta [56]. Vihreän ohjelmiston voidaan ajatella olevan sellainen, jonka suorat ja välilliset kielteiset vaikutukset talouteen, yhteiskuntaan, ihmisiin ja ympäristöön sen kehittämisen, käyttöönoton ja käytön seurauksena ovat mahdollisimman vähäiset tai sillä on myönteinen vaikutus kestävään kehitykseen [13]. Ohjelmiston tarkoituksesta riippuen se voi itsessään olla vihreä (Green In It) ja vihreästi tuotettu. Toisaalta ohjelmiston voidaan ajatella edistävän vihreyttä (Green By It) tai lisäävän tietoutta vihreistä toimintatavoista. [40]

Microsoft on määritellyt kahdeksan ja Green Software Foundation kuusi vihreän ohjelmiston kehittämisen periaatetta. Periaatteet korostavat hiili- ja energiatehokkuutta ja ovat näiltä osin samanlaiset. Alla olevassa listassa on mukaillen yhdistettynä nämä periaatteet.

Hiilitehokkuus	Rakenna hiilitehokkaita sovelluksia.
Energiatehokkuus	Rakenna energiatehokkaita sovelluksia.
Hiilitietoisuus	Käytä sähköä pienimmällä hiili-intensiteetillä.
Laitteistotehokkuus	Rakenna sovelluksia, jotka käyttävät laitteistoa mahdollisimman tehokkaasti.
Laitteiston energiatehokkuus	Maksimoi laitteiston energiatehokkuus.
Verkkotehokkuus	Vähennä datan määrää ja sen kulkemaa matkaa
Mittaus	Mittaa sovelluksen toimintaa.
Optimointi	Lisää yleistä hiilitehokkuutta optimoinnin avulla.
Ilmastositoumukset	Ymmärrä hiilidioksidin vähentämisen merkitys ja mekanismi.

Vihreät laatutekijät määrittelevät, mitä vaatimuksia ohjelmiston on täytettävä ollakseen vihreä, ja mittareilla on tarkoitus mallintaa mitatun ohjelman laatutekijöitä sekä antaa tietoa ohjelmasta. Ensimmäisenä vaatimuksena on, että ohjelmiston suunnittelu-, ohjelmointi, ylläpito- sekä hävittämisprosessi tulee säästää resursseja ja vähentää jätettä. Toisena vaatimuksena ohjelmiston ajaminen tulee säästää resursseja ja tuottaa vähän jätettä. Kolmanneksi ohjelmiston tulee tukea kestävää kehitystä. Näistä vaatimuksista voidaan johtaa kolme vihreää laatutekijää, joita ovat sovellettavuus, suorituskyky sekä kestävyys. Sovellettavuus tarkoittaa sitä, kuinka tehokkaasti sovellus on kehitettävissä, ylläpidettävissä ja uudelleenkäytettävissä. Suorituskyky viittaa siihen, kuinka tehokkaasti sovellus on ajettavissa ja kestävyys viittaa siihen, kuinka hyvin ohjelmisto tukee kestävää kehitystä. Ohjelmiston vihreyttä ajatellessa ei siis riitä, että huomioidaan pelkästään tuotteeseen liittyvät näkökulmat, vaan huomioon täytyy ottaa myös ohjelmistokehitysprosessi. [56] [51].

3.1 Vihreiden käytänteiden implementointi käytäntöön

Ahmad et al. [6] ovat haastatelleet tutkimuksessaan eri yritysten työntekijöitä liittyen vihreiden ohjelmointikäytänteiden toteutumiseen omassa työssään. Tutkimuksesta käy ilmi, että vain osalla yrityksistä on käytössään vihreisiin ohjelmointikäytäntöihin liittyviä ohjeistuksia. Osa haastatelluista ilmoitti, että yrityksellä ei ole tarkkoja ohjeita vihreiden käytänteiden implementoinnista ohjelmointiprosessiin. Tutkimuksessa tulee esille se, että kaikki yritykset eivät myöskään kampanjoi vihreiden käytänteiden esille tuomiseksi. [6] Rajallinen tekninen asiantuntemus ja tietämys, riittämätön infrastruktuuri, asioiden monimutkaisuus ja taloudelliset resurssit ovat esteitä vihreyden implementoinnille [43]. Sugih P. et al. [11] tutkimuksessa tuodaan kuitenkin esille, että kestävän kehityksen mukaisiin työskentelytapoihin kannustetaan myös organisaatiotasolla ja vihreiden käytänteiden huono soveltaminen käytäntöön koskee vain hyvin pientä osaa yrityksistä. Tämän lisäksi työntekijät suhtautuvat pääasiassa positiivisesti vihreisiin sovelluksiin, vaikka niiden käyttämisessä koetaan olevan riskejä. [11]

Vihreään ohjelmistokehitysprosessiin liittyvät käytännöt vaihtelivat. Vaatimusmäärittelyvaiheessa vain osa haastatelluista ilmoitti hyödyntävänsä aikaan, liikkumiseen sekä toimistotiloihin liittyviä säästötoimenpiteitä, etäyhteyksiä, dokumenttien elektronista dokumentointia ja jaettua dokumenttia. Suunnitteluvaiheessa vain osa hyödynsi avoimen lähdekoodin ohjelmistoja, oikeanlaisia käyttöliittymiä, pa-

perin säästämistä ja joustavaa suunnittelua. Osa haastateltujen työpaikoista ei käyttänyt suunnitteluvaiheessa mitään vihreitä käytänteitä. Samoin implementointi- ja testausvaiheessa vihreiden käytänteiden hyödyntäminen oli puutteellista. Koodin optimointia, versiointia, olio-ohjelmointia sekä järjestelmän etätarkistusta hyödynnettiin vain osassa yrityksissä. [6]

Yritysten tulisikin luoda kokonaisvaltainen kestävään kehitykseen ja vihreyteen liittyvä strategia, siihen liittyvät tavoitteet, käytänteet ja toimintatavat sekä aikataulu tavoitteiden saavuttamiseksi [40].

3.2 Vihreä ohjelmistokehitysprosessi

Vihreässä ohjelmistokehityksessä vaatimusmäärittely- ja suunnitteluvaihe korostuvat suhteessa tuotantovaiheeseen. Tämä on ristiriidassa ketterien menetelmien kanssa, jossa painotetaan ohjelmistokehityksen kehittämisvaihetta ja suunnitelmat voivat muuttua useasti prosessin edetessä. Ketterien menetelmien etuna on kuitenkin se, että epäkohtiin niin prosessissa kuin tuotteessakin voidaan puuttua varhaisessa vaiheessa ja tuotteen vihreyttä voidaan parantaa myös prosessin ollessa käynnissä. Oleellista on, että kestävän kehityksen mukaiset toimintatavat integroidaan prosessiin ja suunnitelmiin jo varhaisessa vaiheessa. [12] [8] Yhtä oleellista on kuitenkin parantaa prosessia sen edetessä ja viedä hyväksi havaittuja asioita myös tuleviin projekteihin. Vihreässä ohjelmistokehityksessä on myös tärkeää dokumentoida ohjelmistoon ja prosessiin liittyvät kestävän kehityksen ongelmat, toimenpiteet ja tulokset. Tämäkin on kuitenkin ristiriidassa ketterän ohjelmistokehityksen kanssa, jossa painotetaan kehitystyötä dokumentoinnin sijaan. Ylimääräinen dokumentointi heikentää prosessin vihreyttä, mutta toisaalta se on oleellista, että voidaan todentaa kestävään kehitykseen liittyvät tulokset, raportoida niistä asiakkaalle ja kehittää ohjelmistokehitysprosessin vihreyttä edelleen. Tästä voidaan tehdä johtopäätös, että dokumentoinnin tulisi olla mahdollisimman tiivistä, mutta kuitenkin tarpeeksi kattavaa, jotta se on tarkoituksenmukaista ja ei aiheuta negatiivisia vaikutuksia ohjelmistokehitysprosessiin tai ohjelmistoon.

3.2.1 Vihreän ohjelmistokehitysprosessin menestystekijöitä ja käytänteitä

Hiilijalanjäljen pienentämiseksi organisaatioiden tulisi käyttää vihreitä ja kestäviä käytäntöjä. Vihreän ohjelmistokehitysprosessin merkittävimmät menestystekijät o-

vat vähäiset hiilidioksidipäästöt sekä tehokas resurssien käyttäminen ajan ja laskentaresurssien suhteen. [51] [50]

Ohjelmistokehitysprosessin tehokkuutta voidaan edistää useilla eri keinoilla. Pariohjelmointi on tehokas tapa koodata ohjelmistoja. Pariohjelmointi voi tuoda lisäaikaa yksinkertaisten tehtävien suorittamisessa ja toisaalta helpottaa monimutkaisten tehtävien ratkaisemista laadukkaammin. Tiimin tulisi myös kehittää uudellenkäytettävää koodia. [19] [50] [52]

Tuotteen luovutus asiakkaalle tulisi aikatauluttaa ja seurata kehitysprosessin nopeutta, jotta ohjelmisto tulee luovutettua ajoissa. Sprintit/iteraatiot tulisi pitää lyhyinä ja prosessin aikana tulisi huolehtia siitä, että korjattavat viat eivät kasaannut. Ohjelmistokehitykseen tulisi käyttää olemassa olevia työkaluja. [50]

Hyvä ja tehokas viestintä on yksi vihreän ja ketterän ohjelmistoprojektin menestystekijöistä. Tiimin keskinäistä sekä tiimin ja asiakkaan välistä viestintää ja yhteistyötä voidaan edistää esimerkiksi valmentamalla tiimiä ennen projektin alkamista. Viestintää voidaan myös tehostaa kasvokkain tapahtuvilla tapahtumilla sekä verkon välityksellä tapahtuvalla viestinällä. Tehokkaan viestinnän näkökulmasta tiimin koko tulisi pitää mahdollisimman pienenä. Epävirallisten tapaamiset sekä sosiaaliset aktiviteetit edistävät niin ikään tiimin välistä sekä tiimin ja asiakkaan välistä kommunikointia. Positiivinen työympäristö ja riittävä tuki työntekijöille luo edellytyksiä tiimin sisäiselle yhteistyölle sekä tehokkaalle ohjelmistokehitykselle ja lisää siten tuottavuutta. [50]

Minimaalin dokumentointi on yksi kriittisistä menestystekijöistä vihreässä ketterässä ohjelmistokehityksessä. Minimaalisella dokumentaatiolla tarkoitetaan sitä, että ohjelmistokehityksessä painotetaan koodin kirjoittamista raskaan dokumentaation sijasta. Ohjelmistokehitysprosessina aikana tulisi järjestää tapaamisia asiakkaan ja tiimin välillä, jotta saavutettaisiin mahdollisimman hyvä ymmärrys projektin monimutkaisuudesta ja vähennettyä turhaa dokumentaatiota. Raskasta dokumentaatiota voidaan pienentää epävirallisella viestinnällä tiimin jäsenten kesken sekä tehokkaalla ohjelmistokehitystä koskevien tietojen jakamisella. [50] Minimaaliseen dokumentaatioon liittyy kuitenkin ongelmia, jotka saattavat haitata myöhemmissä vaiheissa ohjelmistokehitysprosessia tai ohjelmiston elinkaarimallia. Dokumentaation tulisi olla riittävää ja laadukasta. Minimaalinen dokumentointi olisikin hyvä yhdistää dokumentoinnin automatisointiin erilaisilla työkaluilla. [57] Tästä voidaan päätellä, että dokumentoinnin tulisi olla optimoitua, jotta dokumentaation puute ei haittaa ohjelmistokehitystä muissa vaiheissa. Toisaalta pitää huolehtia siitä,

että turha dokumentaatio ei aiheuta esimerkiksi sähköistä jätettä ja hukkatyöaikaa, joka heikentää ohjelmiston vihreyttä.

Monimuotoinen suunnitellut tarkoittaa asiakkaan tarpeiden mukaisesti suunniteltua, yksinkertaista ja käyttäjäystävällistä ohjelmistoa ja sen voidaan ajatella olevan yksi kriittisistä vihreän ohjelmiston menestystekijöistä. Ketterän ohjelmistokehitysprosessin mukaisesti suunnitelmia päivitetään asiakkaan toiveiden mukaiseksi ja validoidaan hyväksymistestauksen avulla. Tämän takia on oleellista, että asiakas on tiiviisti mukana ohjelmiston kehittämisessä koko prosessin ajan. Komponenttipohjaisella kehitysstrategialla voidaan lisätä prosessin vihreyttä, koska se mahdollistaa ohjelmiston osien uudelleenkäytön ja vähentää ajan käyttöä sekä kustannuksia prosessissa. Koodin refaktoroinnin avulla voidaan muuttaa ohjelmiston sisäistä rakennetta esimerkiksi komponenttipohjaisen ohjelmiston selkeyttämiseksi ja myös tehostamiseksi. Suunnitellut moduulit tulisia olla vain vähän riippuvaisia toisistaan, jotta ohjelmiston rakenne voidaan pitää mahdollisimman yksinkertaisena. Yksinkertaisen ja tarkoituksenmukaisen ohjelmiston suunnittelussa tulisi olla mukana kokeneita ohjelmistokehittäjiä. [50]

Vihreän ohjelmistokehitysprosessin tukena voidaan hyödyntää tarkistuslistoja [13] [12]. Tarkistuslistat ovat yksinkertainen ja helposti toteutettavissa oleva tapa varmistaa vihreiden näkökulmien huomioon ottaminen ohjelmistokehitysprosessin jokaisessa vaiheessa. Tarkistuslistat voidaan osoittaa tietyille työntekijärooleille erikseen, jossa otetaan huomioon juuri kyseisen työntekijän työtä koskevat näkökulmat. Näitä rooleja voivat olla esimerkiksi web-kehittäjä, sovellusarkkitehti jne. [13]

Ohjelmistokehitysprosessista voidaan saada merkittävästi vihreämpi toteuttamalla usein pieniä muutoksia työskentelytapoihin. Esimerkiksi tietokoneet ovat usein päällä tuhlaamassa energiaa myös silloin, kun ne eivät ole aktiivisessa käytössä. Tietokoneen käyttämän virran lisäksi ylimääräistä energiaa kuluu tietokoneen viilennykseen. Yhden koneen sammuttaminen ja siitä saatavat säästöt ovat pieniä, mutta suuremmassa mittakaavassa säästöt ovat merkittäviä. Mikäli tietokoneen sammuttaminen ei tule kysymykseen, tulisi käyttää näytönsäästäjiä. Tyhjä näytönsäästäjä säästää enemmän energiaa kuin sellainen, jossa on liikettä. Näiden lisäksi kehitystyössä voidaan hyödyntää kevyitä asiakaspäätteitä, jotka käyttävät vain viidenneksen energiaa tavalliseen pöytätietokoneeseen verrattuna. [40]

Paperittomat työskentelytavat ovat oleellinen osa vihreitä toimintatapoja [6] [13] [51]. Myös etätyöskentely ja etäyhteyksien hyödyntäminen vähentää suoraan mat-

koista aiheutuvia päästöjä, mutta se myös säästää matkoihin kuluvaa työaikaa [61] [13].

3.2.2 Ohjelmistokehitysprosessin vihreyden mittaaminen ja raportointi

Ohjelmistokehitysprosessin vihreyden mittaamisen tarkoituksena on arvioida itse prosessin vihreyttä. Prosessin arvioiminen tapahtuu prosessin jokaisessa vaiheessa. Prosessin vihreydestä tulisi olla vastuussa tähän rooliin nimetty henkilö. Prosessin eri vaiheissa kerättävien tietojen tulisi olla riittävän kattavaa, jotta prosessin lopussa voidaan raportoida luotettava hiilijalanjälkilaskelma. Tarkoituksena on, että prosessin vihreyden arviointi aloitetaan jo siinä vaiheessa, kun ohjelmistoa aletaan ideoimaan ja jatketaan kehitysprosessin jokaisessa vaiheessa joko toteuttaen, dokumentoiden tai arvioiden tuloksia. [12] Prosessin vihreyden lisäksi tuotteen suoria ja epäsuoria ympäristövaikutuksia tulisi mitata koko prosessin ajan [13].

Dick et al. [12] ehdottavat vihreyden arviointiin ja katsaukseen tarkoitettua tapaamista (A Review and Preview) noin kahden kolmasosaa iteraation jälkeen. Tämän tapaamisen tarkoituksena on tarkastella kestävän kehityksen osalta tehtyä työtä ja mahdollistaa iteraation aikana löydettyjen suunnittelu- ja toteutusvirheiden korjaaminen saman iteraation aikana. Tätä varten tiimin tulisi suorittaa arviointeja ja katselmuksia. Vihreyden arviointi ja katsaus -tapaamisen sekä prosessin arviointi yhdistetään prosessin vihreyttä dokumentoivassa päiväkirjassa (Sustainability Journal). Tämä on lyhyt dokumentti prosessin arvioinnista sekä parannusehdotuksista, jonka päivittäminen on sekä kehitystiimin, että kestävän kehityksen johtajan vastuulla. Vihreyttä koskevat esitykset tulisi integroida Sprint Reviw -kokouksiin, jolloin tuotteen omistaja hyväksyy tai hylkää kehitetyt ominaisuudet ja siten myös toimenpiteet, joilla tuotteen vihreyttä on paranneltu. [12] [50]

Iteraatiokierroksen lopussa raportoidaan kestävää kehitystä koskevista toimenpiteistä. Tavallisesti näitä ovat tuotetta koskevat ongelmat, ratkaisut sekä näiden onnistuminen. Prosessin vihreydestä vastuussa henkilö raportoi prosessista syntyneet vaikutukset, joita ovat esimerkiksi liikematkojen päästöt, kulutetut henkilöpäivät jne. Projektin lopussa hän myös kokoaa projektia koskevat tulokset, kuten prosessin vihreyttä dokumentoivaan päiväkirjan ja muut väliesitykset, yhteen ja julkaisee tulokset asiakkaalle. [12]

Kestävän kehityksen retrospektiivi tulisi toteuttaa ohjelmistoprojektin päättyessä ennen viimeisen sprintin katsauksen päättymistä. Muista ketterän kehityksen vaiheista poiketen kestävän kehityksen retrospektiiviä ja sprintin retrospektiiviä ei

tule yhdistää, koska näillä on erilainen asema prosessin aikajanassa sekä painopisteen suhteen. Kestävän kehityksen retrospektiivissä keskitytään tiimin kokemukseen kestävästä kehityksestä sekä viemään hyväksi havaittuja sekä toimivia käytänteitä seuraaviin projekteihin. Tulokset ja asiakkaan antama palaute voidaan kirjata esimerkiksi tietokantaan tai prosessikäsikirjaan. [12] [50]

Sovellettavuuden mittareita ovat mm. hiilijalanjälki, energian kulutus, matkustamisen määrä ja hukkaan kuluvien resurssien määrä. Hiilijalanjälki määrittää, kuinka paljon hiilidioksidia ohjelmistokehitys-, hallinta- tai ylläpitovaihe tuottaa. Tämä on tärkein vihreä mittari ja lopulta kaikki vihreät tekijät pitäisi laskea sen avulla. [56] Hiilijalanjälki on kuitenkin ongelmallinen siinä suhteessa, että sitä voidaan manipuloida esimerkiksi valitsemalla uusiutuvia energianlähteitä sähkön tuottamiseen. Hiilijalanjäljen pienentämistä tällä tavoin voidaan kuitenkin verrata oireiden hoitamiseen, se ei paranna varsinaista tautia, joka on liika energiankulutus. [61]

Sovellettavuus määrittelee kuinka hyvin ohjelmistojen kehittämisprosessit, ylläpito ja ohjelmistojen käytöstä poistaminen tukee kestävää kehitystä. Sovellettavuuden laatutekijöitä noudattava ohjelmistokehitysprosessi tukee kestävää kehitystä, minimoi resurssitarpeen ja tuottaa mahdollisimman vähän jätettä. Sovellettavuus painottaa ohjelmistosuunnitteluprosessia lopputuotteen sijaan. Se on helposti mitattavissa ja siihen on valmiiksi olemassa hyviä tekijöitä ja mittareita. Tekijät liittyvät tässä ihmisten käyttäytymiseen. Tuloksena on usein jokin absoluuttinen arvo, joka voidaan suoraan lisätä prosessin tuottamaan hiilijalanjälkeen. Koska sovellettavuus on helppo mitata, ohjelmistojen vertailu on käytännöllistä. [56]

Energia määrittelee, kuinka paljon energiaa kulutetaan kehitysvaiheiden aikana. Hiilijalanjälki-mittari ottaa huomioon sen, miten energia tuotetaan, joten sitä ei huomioida tässä mittarissa. Hiilijalanjälki-mittari on mittareista tärkein, koska sitä voidaan käyttää epäsuorien mittareiden perustana. Matkustamiseen liittyvä mittari määrittelee, kuinka paljon ohjelmiston kehitysprosessin aikana matkustetaan. Matkustaminen vie aikaa ja vaatii resursseja, joten sen seuraaminen on tärkeää. Hukka määrittelee, kuinka paljon resursseja kuluu toimintoihin, jotka eivät tuota näkyvää arvoa ohjelmiston käyttäjille. Hukka voi olla jotain fyysistä, hukkaenergia tai prosessiin liittyvää. Prosessihukka on toimintaa, joka vaatii resursseja, mutta ei tuota mitään arvokasta. Esimerkiksi näitä ovat käyttämättömät kehitysajat ja odotusjonot. [56] Tiimit voivat mitata ja arvioida energiatehokkuuttaan esimerkiksi laitteiston tehomittarilla [12].

Edellisistä kappaleista voidaan huomioida se, että ohjelmiston kestävyyden tai

vihreyden arvioimiseksi ei riitä pelkästään energiankäyttöön liittyvät mittaukset. Kestävyyden arvioimiseksi on kehitetty useita erilaisia menetelmiä, jotka eroavat toisistaan siinä kuinka hyvin ne huomioivat erilaiset näkökulmat kestävyyttä tai vihreyttä arvioitaessa. [52]

3.3 Vihreän ohjelmiston elinkaarimalli

Seuraavissa alaluvuissa vihreää ohjelmistoa käsitellään elinkaarimallin näkökulmasta. Elinkaarimallista on otettu mukaan vaatimusmäärittely-, suunnittelu-, toteutus-, testaus- ja ylläpitovaihe.

3.3.1 Vaatimusmäärittely

Vaatimusmäärittelyvaihe voidaan katsoa edeltävän suunnitteluvaihetta ja tuotteen kehitystä. Scrumissa, joka on yksi ketterä prosessinhallintamenetelmän muoto, sisältää Product Backlog -artefaktin, joka on listaus siitä mitä ohjelmiston parantamiseksi tarvitaan. [53] Asiakkaan tarpeet ja toiveet voivat kuitenkin muuttua tuotteen kehityksen aikana ja ketterissä menetelmissä pyritään vastaamaan myös asiakkaan muuttuneisiin vaatimuksiin [50]. Ketterissä menetelmissä vaatimusmäärittelyn voidaankin katsoa toistuvan useaan kertaan prosessin aikana ja kertaalleen läpikäytävää vaatimusmäärittelyvaihetta ei ole.

Ohjelmistokehitysprojektin tavoitteena on usein paras mahdollinen käyttäjäkokemus myynnin edistämiseksi, ja energiatehokkuus ei tällöin aina ole ensisijainen huolenaihe. Ongelmana on myös se, että kaikki vaatimusmäärittelyihin liittyvät yksityiskohdat eivät sisällä kaikkia kestävyyteen liittyviä ulottuvuuksia. Esimerkiksi käyttöönottovaatimukset voivat sisältää vain tekniikkaan liittyviä vaatimuksia. Toisaalta ohjelmistolle asetetut vaatimukset voivat olla ristiriidassa keskenään ja esimerkiksi resurssit asettavat rajoitteita sille voidaanko ympäristöystävällisemmät, mutta kalliimmat ratkaisut toteuttaa ohjelmistokehityksessä. Ympäristöystävällisten ratkaisujen vaikuttavuutta on myös vaikea mitata ja ympäristönsuojelulle ei voida myöskään suoraan asettaa mitään rahallista arvoa. Tämän takia vihreiden tavoitteiden saavuttamiseksi täytyisi määritellä tarkat toimenpiteet ja tulisi perustella esimerkiksi yrityksen hallituksen asettamilla strategisilla tavoitteilla tai yrityksen sitoutumisella kestävän kehityksen ohjelmiin. [45] [52] Esimerkki kestävän kehityksen ohjelmasta World Business Council for Sustainable Development:n määrittele-

mä Visio 2050-ohjelma [63].

Energiankäyttöön liittyvät vaatimusmäärittelyt on esitetty usein toiveina tarkkojen vaatimusmäärittelyjen sijaan ja energiankäyttövaatimukset on saatettu ilmaista muilla kuin energiankäyttöön liittyvillä termeillä. Epätarkasti ilmaistut ja vaillinaiset tavoitteet voivat yhdessä johtaa siihen, että energiankulutusta ei huomioida kaikissa ohjelmiston toiminnoissa. Esimerkiksi energiankulutus pyritään optimoimaan ohjelmiston joutokäynnin ajalle, mutta suorituksen aikainen energiankulutus saattaa jäädä huomioimatta. Vaatimusten tulisi siis auttaa ohjelmistokehittäjiä hahmottamaan kuinka paljon energiaa on järkevää käyttää jonkin tietyn toimenpiteen suorittamiseen. [36]

3.3.2 Suunnittelu

Suunnitteluvaihe voidaan katsoa edeltävän toteutusvaihetta. Scrumissa jokainen sprintti sisältää sprintin suunnittelukokouksen, joka voidaan katsoa olevan osa suunnitteluvaihetta [53]. Säännöllisten suunnittelukokousten pitäminen on vihreän ohjelmistokehitysprosessin yksi menestystekijöistä. Suunnittelukokouksissa voidaan puuttua esimerkiksi edellisen kerran jälkeen havaittuihin epäkohtiin, tehostaa prosessia tai lisätä suunnitelmaan toimintoja, joilla pyritään parantamaan ohjelmiston laatua ja vihreyttä. [50] [51]

Ohjelmistosuunnittelun eri vaiheissa on useita suunnittelutoimintoja, kuten käsitteellinen-, looginen-, fyysinen-, tietorakenne- ja algoritmisuunnittelu, joilla voidaan edistää ohjelmiston vihreyttä. Vaatimukset tulisi suodattaa vihreän arvioijan avulla. Esimerkiksi "Green Tracker-energianarviointiohjelmistolla voidaan mitata ohjelmiston energiankulutusarviota. Arvioinnin aikana tulisi tarkastella ohjelmiston käyttöikää vaatimusten keruuvaiheessa nykyisen ja tulevaisuuden näkökulmasta. [51] [50]

Verdecchia et al. [61] ja Ahmad [6] painottavat, että kestävän kehityksen näkökulmasta ohjelmistoja tulisi alkaa suunnitella ja toteuttaa siten, että ne olisivat mahdollisimman joustavia. Suunnittelijoiden tulisi huomioida erityisesti se, että uudet ohjelmistoversiot eivät kuluta enempää energiaa, muistia tai kaistanleveyttä kuin vanha. Yksinkertainen keino edistää sovelluksen kestävyyttä on varmistaa ohjelmistojen ydintoimintojen toimiminen myös vanhemmissa laitteistoissa. [61] [6] [13]

IoT-laitteiden (Internet of Things -laitteiden) suunnittelussa ja toteutuksessa on paljon hyviä käytänteitä, joita voidaan hyödyntää myös muussa ohjelmistosuunnittelussa. Esimerkiksi tulisi välttää sellaisia ratkaisuja, jotka kuluttavat virtaa jatku-

vasti, valita tehokkaat algoritmit, välttää jatkuvaa tietoliikenneyhteyttä, tarkkuutta sekä ylimääräistä muistin käyttöä. [61] [6]

3.3.3 Toteutus

Ohjelmiston kehittämisen aikainen tehokas koodaus on yksi tärkeimmistä menestystekijöistä vihreän ohjelmiston tekemisessä. [51] Ahmad et al. [6] tutkimuksessa käy kuitenkin ilmi, että kehitystyön aikaista kestävän kehityksen mukaista implementointia ohjelmistoon tai prosessiin ei välttämättä ole ollenkaan.

Manotas et al. [36] mukaan ohjelmistokehittäjillä on usein huono mielikuva siitä, kuinka paljon heidän kehittämänsä ohjelmisto kuluttaa energiaa. Sen takia ohjelmistokehittäjillä tulisi olla käytettävissä energiankulutusta profiloivia työkaluja. Tällaisen työkalun tulisi ottaa huomioon myös se, miten kehitettävä ohjelmisto käyttäytyy muiden sovellusten kanssa. Kehittämisen aikana ohjelmiston energiatehokkuutta voidaan parantaa mm. staattisen koodianalyysin avulla. Ohjelmistokehittäjän tulisi myös osata käyttää ohjelmointikielen ominaisuuksia siten, että ohjelmisto toimii optimaalisesti energiankulutusta arvioitaessa. [36]

Ohjelmistokehityksen aikaisista toimenpiteistä ja tekniikoista kerrottaan tarkemmin luvussa 3.4 Vihreä ohjelmisto.

3.3.4 Testaus

Vesiputousmallissa testausvaihetta voidaan ajatella omana erillisenä vaiheena kehitystyötä. Ketterissä menetelmissä kuten Scrumissa testausta toteutetaan säännöllisesti esimerkiksi jatkuvan integraatioympäristön avulla. Tähän testisuoritukseen voidaan integroida energia- ja suorituskykymittaus, joka toimitetaan yhdessä muiden testitulosten kanssa kehittäjille. [12] Vihreässä ohjelmistokehityksessä on myös oleellista tehdä hyväksymistestaus asiakkaan vaatimusten vahvistamiseksi [50].

Verdecchia et al. [61] kertovat artikkelissaan projektista, jossa raakaan voimaan perustuvaa testausta muokattiin testistrategian avulla kohti älykästä testausta. Raakaan voimaan perustuva testaus korreloi vahvasti energiankulutuksen kanssa. Ohjelmistoprojekteissa testaaminen onkin merkittävässä osassa ohjelmistoprojektin energiankulutuksessa. Energiankulutuksen vähentämiseksi ja testauksen laadun parantamiseksi olisi tärkeää suorittaa testitapaukset testausstrategian mukaisesti. [61]

Projektissa testausprosessi jäsenneltiin systemaattisesti sekä otettiin käyttöön testausmenetelmiä, kuten esimerkiksi staattinen koodianalyysi. Staattisessa koodiana-

lyysissä pyritään tunnistamaan ohjelmiston osia tai alueita koodista, jotka vaativat enemmän matalan tason testausta. Testitapaukset ja testien hallinta instrumentointiin testin tehokkuuden tunnistamiseksi. Tehokkuutta pyrittiin parantamaan kohdistamalla testausta aiemmin kattamattomiin koodisegmentteihin. Testattavuutta pyrittiin parantamaan testivetoisen kehityksen (TDD) ja testilähtöisen vaatimussuunnittelun (TDRE) avulla, joka määrittelee toiminnot testitapauksiksi. Näillä toimenpiteillä pystyttiin poistamaan toisteisuutta testauksessa ja parantamaan tehokkuutta. Testilähtöinen ohjelmistokehitys ja testilähtöinen vaatimussuunnittelu helpottavat regressiotestausta ja tekevät siitä tehokkaampaa. [61]

Projektin toisessa vaiheessa otettiin käyttöön riskilähtöinen testaus. Tällä pyritään vastaamaan kysymyksiin siitä, miten järjestelmän toimivuus, turvallisuus ja luotettavuus voidaan varmistaa, jos ohjelmiston ajonaikainen toimintahäiriö vaikuttaa johonkin ohjelmiston toisen osan alikomponenttiin tai jos päivityksiä tehdään vain osaan ohjelmistoa. Riskilähtöinen testaus arvioi funktioiden riippuvuuksia sekä valkoinen laatikko- ja musta laatikko -testauksena (White-box and Black-box -testing). Projektin viimeisessä vaiheessa otettiin käyttöön älykäs testaus tekoälyn kanssa. Älykäs testaus auttaa määrittelemään mitkä testitapaukset on toteutettava ja missä määrin. Älykäs validointi auttaa testitapausten valinnassa ja luomisessa. Projektin tuloksena testaamiseen käytettäviä resursseja saatiin vähennettyä ja testien tehokkuutta parannettua. [61]

3.3.5 Ylläpito

Ohjelmistot, joita voidaan käyttää pidempään ilman muutoksia tai parannuksia, vähentävät ylläpitotoimia ja puolestaan auttavat pienentämään hiilijalanjälkeä. Kaikki ohjelmiston ylläpitotoimet katsotaan tuoreeksi kehitystyöksi ja johtavat koko SDLC-prosessin toistamiseen. Siten ohjelmistoja, joiden käyttöikä on pidempi, voidaan pitää myös kestävänä ohjelmistona. Termi kestävä koskee sekä ohjelmiston pidempää elinikää että vihreämpiä näkökohtia. [3]

Ylläpitovaiheessa huomioidaan ohjelmiston käyttämisestä ja ylläpitämisestä syntyvät suorat ja epäsuorat vaikutukset. Ylläpitovaiheeseen liittyvät asiat tulee kuitenkin ottaa huomioon jo ohjelmistoa suunniteltaessa ja kehitettäessä. Tässä vaiheessa tulisi huomioida esimerkiksi järjestelmän tuki- ja konfigurointitehtävät. Loppukäyttäjille voidaan myös tarjota energiansäästöohjeita. Ohjelmiston elinkaaren viimeisessä vaiheessa huomioidaan ne asiat, jotka ovat merkityksellisiä, kun ohjelmistoa poistetaan käytöstä. Tällöin tulisi päättää mitkä tiedostot säästetään, voidaanko nii-

tä muuttaa uusiin tiedostomuotoihin vai voidaanko ne poistaa kokonaan ja säästä siten muistiresursseja. [13]

Ohjelmiston ylläpitovaiheeseen kuuluu myös koodin refaktorointi, joka vihreässä asiayhteydessä tarkoittaa lähdekoodin energiatehokkuuden optimointia muuttamatta lähdekoodin rakennetta. Kaikki olemassa olevat refaktorointitekniikat eivät kuitenkaan vaikuta ohjelmiston energiatehokkuuteen. [18]

Verdecchia et al. tutkimus [61] tarjoaa näkökulman käytön aikana syntyvän hiilijalanjäljen pienentämiseen sillä, että käyttäjät ovat myös vastuussa ja pystyvät vaikuttamaan omalla toiminnallaan tähän. Yksinkertainen esimerkki on mainosten estäminen, jolla voidaan vähentää energian kulutusta datakeskuksissa, tietoliikenneväylillä sekä omassa tietokoneessa. Ohjelmiston kehittäjän vastuulla on mahdollistaa energiaa säästävät ja hiilijalanjälkeä pienentävät toiminnot ja tehdä niistä helppokäyttöisiä sekä informatiivisia. Koulutusohjelmien vastuulla on, että tulevaisuuden tekijöillä on tarpeeksi tietotaitoa toteuttaa energiatehokkaita sekä vihreitä ohjelmistoja.

3.4 Vihreä ohjelmisto

Seuraavissa luvuissa käydään läpi ohjelmiston rakentamiseen liittyviä käytänteitä, joilla pyritään edistämään näitä vihreän ohjelmiston kehittämisen periaatteita.

3.4.1 Ohjelmiston vihreyden mittaaminen

Ketterässä kehityksessä on tavallista, että tuotteita kehitetään testilähtöisesti. Tavallisesti on totuttu siihen, että testaus kohdistuu muihin ohjelmiston laatutekijöihin kuin vihreyteen ja siksi kehittäjien tulisi integroida testipakettiin myös testejä energiatehokkuuden arvioimista varten. [12]

Aggrawal et al. [5] kertovat tutkimuksessaan, että järjestelmäkutsuprofiili korreloi ohjelmiston energiankulutuksen kanssa. Ohjelmistokehittäjät voivat hyödyntää järjestelmäkutsuprofilointia ohjelmistoon tehtyjen muutosten yhteydessä arvioimaan ohjelmiston energiankulutuksen muutoksia. Järjestelmäkutsuprofilointiin on olemassa myös yksinkertaisia työkaluja, kuten Aggrawal et al. [4] kehittämä GreenAdvisor.

Hyvä ohjelmisto säästää resurssejaan ja aika sekä vähentää hukkaa. Tehokkuus määrittelee, miten ohjelmisto käyttäytyy resurssien säästämisessä ja tuhlauksen vält-

tämisessä. Tehokkuuden mittareita ovat muun muassa CPU-intensiteetti, muistin käyttö, perifeerinen intensiteetti, joutokäynti sekä heijastuskyky. CPU-intensiteetillä mitataan kuinka monta CPU-jaksoa ohjelmisto kuluttaa. Jokaisen syklin vaatima energia on mitattavissa ja mukautettavissa muihin mittareihin. Muistin käytöllä voidaan seurata esimerkiksi päämuistin kulutusta. Tämän lisäksi seurataan, kuinka muistia käytetään. Erittäin vähän energiaa kuluttavassa ympäristössä on hyödyllistä minimoida muistin kulutus ja nollata käyttämättömät tavut. Tällä säästetään energiaa siten, että nollattu bitti on edullisempaa päivittää kuin asetettu bitti. [56]

Perifeerisellä intensiteetillä seurataan sitä, kuinka paljon oheislaitteita käytetään. Tämä voidaan arvioida esimerkiksi laskemalla, kuinka monta pyyntöä ohjelmisto tekee oheislaitteille ja mitä resursseja tarvitaan. Toinen yksinkertainen tapa on laskea oheislaitteen energiatarve. Tarkoituksena on, että suorittimien syklit olisivat jollain tavalla hyödyllisiä ja auttaisi ohjelmistoa saavuttamaan tavoitteensa. Tyhjäkäynti ei kuitenkaan koskaan ole hyödyllistä ja sillä hukataan resursseja. Joutokäynnillä siis seurataan, kuinka paljon ohjelmisto on käyttämättömänä. Joutokäynnin minimoimiseen on kehitetty muun muassa IT-palvelinratkaisuja. Heijastuskyvyllä mitataan kuinka ohjelmisto vaikuttaa epäsuorasti sen toimialueeseen. Heijastuskyky on ehkä tärkein tehokkuuden tekijä, koska siitä syntyvän jätteen määrä voi olla tuhoisa. Esimerkiksi muistiin perustuva ohjelmisto käyttää resursseja yleensä silloin kun tietokone käynnistetään. Tämä hidastaa tietokoneen käynnistystä ja tietokoneen käyttäjä jättää herkemmin tietokoneen päälle välttääkseen odottelua. Tämä lisää suorasti energiankulutusta sekä huonontaa tehokkuutta. Näillä tekijöillä arvioidaan, kuinka ohjelmistot vähentävät jätettä. Tehokkuusmittareiden tulkinta ei kuitenkaan ole yksinkertaista. [56]

Kestävyyttä mitattaessa halutaan tietää mikä ohjelmistojen arvo on kestävässä kehityksessä sekä kuinka ohjelmisto tukee hävikin vähentämistä. Kestävyys on suhteellinen tekijä, joten kahden ohjelmiston kestävyyttä ei voi verrata, jos ne eivät ole samanlaisissa ohjelmistojärjestelmissä. Kestävyys on vaikeinta mitata sen abstraktin luonteen takia. [56] Kestävä kehitys voidaan määritellä tarkoitukseen sopivuudella, reduktiolla sekä kauneudella. Se voidaan myös jakaa myös ekonomiseen, sosiaaliseen ja ympäristölliseen ulottuvuuteen. [24] Tarkoitukseen sopivuus määrittää kuinka ohjelmisto auttaa järjestelmää saavuttamaan tavoitteet. Reduktio määrittää kuinka ohjelmisto tukee jätteen vähentämisessä. Kauneus määrittelee ohjelmiston arvon kestävässä kehityksessä. Näillä tekijöillä ei kuitenkaan ole mahdollista saada absoluuttisia mittareita. Tekijät ovat riippuvaisia järjestelmästä, jossa ohjelmis-

toa suoritetaan sekä toimialueesta, jossa järjestelmää käytetään. Tämän takia näiden tekijöiden mittaaminen on toimialuekohtaista ja jokaiselle järjestelmälle täytyy määritellä vertailujärjestelmä. [56]

3.4.2 Laitteisto

Verdecchia et al. [61] kirjoittavat artikkelissaan, että kaiken kaikkiaan tuotteet tulisi suunnitella joustavaksi tulevaisuutta ajatellen. Yksinkertaisinta on varmistaa, että ohjelmiston ydintoiminnot toimivat edelleen vanhemmissa laitteissa. Laitteiden ikääntyminen on väistämätöntä, mutta tämänhetkinen tilanne, jossa laitteiston odotetaan vanhenevan todella lyhyessä ajassa, on kestämätöntä tulevaisuutta ajatellen. IoT-laitteiden ohjelmoijat ovat ratkaisseet tämän ongelman siten, että he eriyttävät ohjelmiston laitteistosta. Tämä mahdollistaa esimerkiksi tietoturvakorjaukset sekä ohjelmisto päivitykset ilman laitteistoin uusimista. Usein päivitykset tapahtuvat ilmateitse, kuten IoT-laitteiden luonteeseen kuuluu. Ilmateitse tapahtuvat päivitykset mahdollistavat myös tietoturvapäivitykset ja virheiden korjaukset ilman uusia laitteistoja. Myös pilvipalvelujen käyttö vähentää tarvittavien laitteistojen määrää ja tarvetta päivittää laitteistoja. [2].

Osa isoista laitevalmistajista on alkanut strategisesti kehittää laitteita hiilineutraaliin suuntaan. Asiakkaat voivat arvioida tuotteiden ominaisuuksia esimerkiksi standardien ja tuoterekisterien avulla. Laitevalmistajat ovatkin alkaneet valmistaa vihreämpiä tietokoneita esimerkiksi käyttämällä myrkyttömiä vaihtoehtoja materiaaleille sekä tekemällä niistä vähemmän energiaa kuluttavia. Käyttöikää on pyritty pidentämään tekemällä tietokoneista päivitettävämpiä. Ydinprosessoreiden määrän lisääminen sirun taajuuden lisäämisen sijaan säästää energiaa ja lisää myös tietokoneen suorituskykyä. Esimerkiksi 15 % säästö taajuudessa vähentää virrankulutusta 50 %. Myös välimuistin jakaminen segmentteihin ja niiden käyttäminen vain tarvittaessa säästää energiaa. Flash-muistia voidaan käyttää välimuistina kiintolevyllä, pöytäkoneilla ja palvelimilla. Näytöissä energiaa voidaan säästää myös käyttämällä tummempaa taustavaloa. [40]

Edelleen laitteistojen kehittyessä ja tutkimuksen edetessä on mahdollista, että tulevaisuudessa otetaan käyttöön non-Von Neumann -tekniikoita, kuten neuromorfisen laskennan. Tämä mahdollistaisi laskennan suorittamisen murto-osalla nykyisistä energiakustannuksista. [61]

3.4.3 Säästävät ohjelmistostrategiat

Säästävillä ohjelmistostrategioilla pyritään minimoimaan virrankulutus, joten se voidaan katsoa olevan tärkeä osa vihreää ohjelmistokehitystä. [51] Kehittäjien tulisi adaptoida ohjelmointitekniikoita, jotka ovat mahdollisimman tehokkaita tiedon varastoinnin, laskennan sekä verkkoresurssien käytön suhteen. Tämä voi tarkoittaa esimerkiksi ohjelmiston koordinointia laitteiston kanssa, tyhjäkäyntitehokkuuden optimoimista tai laiskaa initialisointia (Lazy Initialization). Laskennan ei tarvitse kaikissa tapauksissa olla tarkkaa ja oikea-aikaista. Energiaa voidaankin säästä myös viivästyttämällä sitä ohjelmiston vaatimusten niin salliessa. [61] [49] [40] [36]

Energiaa voidaankin säästä myös approksimoimalla laskennan tarkkuutta. [36] Mitra et al. [38] selvittivät tutkimuksessaan ohjelmiston laskennan approksimoimista jakamalla ohjelmiston toiminnan vaiheisiin ja tutkimalla yksittäisten vaiheiden vaikutusta palvelunlaatuun, nopeuteen sekä energiankulutukseen. Tällaisen vaiheittaisen optimoinnin avulla pystyttiin säästämään energiaa ilman, että approksimoiminen vaikutti liikaa ohjelmiston laatuvaatimuksiin.

Ohjelmiston käytöksellä on merkittävä vaikutus alustaan sisäänrakennettujen energiansäästöominaisuuksien tehokkuuteen. Ohjelmiston energiatehokkuutta on laskentatehokkuus. Laskentatehokkuus tarkoittaa sitä, että ohjelmisto tulee rakentaa tehokkailla algoritmeilla, hyödyntämällä rinnakkaislaskentaa sekä vektorointia. [55] [49] [25] Kambadur ja Kim [25] totesivat tutkimuksessaan rinnakkaislaskennan olevan toimiva tapa vähentää ohjelmiston energiankulutusta. Tuloksia heikensi kuitenkin se, että osa testatuista ohjelmista soveltui huonosti rinnakkaislaskentaan. Energiansäästöjä saavutettiin kuitenkin myös huonosti soveltuvien ohjelmien rinnakkaistamisessa.

Datatehokkuudella tarkoitetaan sitä, että datan liikkuminen operoinnin aikana pidetään mahdollisimman vähäisenä. Tämä tarkoittaa I/O-kutsujen (Input/Output -kutsujen) optimointia sekä pollauksen eli laitteiden tilan tarkastamisen välttämistä. Data tulisi siirtää massaoperaationa (Bulk Operation). Algoritmien lisäksi muistihierarkian ja datastruktuurien hyvä suunnittelu sekä välimuistin tehokas käyttäminen ovat tärkeässä roolissa. Kolmas kategoria eli kontekstitietoisuus tarkoittaa sitä, että sovellus pystyy tarkkailemaan ympäristöään ja reagoimaan ympäristön muutoksiin. [61] [49]

Suurin ohjelmistokeskeisen Green IT -lähestymistavan mahdollistanut teknologia on virtualisointi. Sekä suuret että pienet organisaatiot hyödyntävät virtualisointia lähinnä palvelimiensa ja työasemiensa kustannusten alentamiseksi. Epähuo-

miossa tämä on myös vaikuttanut paljon kestävään ohjelmistoympäristöön. Siksi ohjelmiston käyttötapa on tärkeä ohjelmiston kestävyyden kannalta. [3] Myös tekoälyllä ja koneoppimisella voidaan saada säästöjä siten, että tunnistetaan eri datankäyttöprofiileja ja optimoidaan palvelimien käyttöä entistä tehokkaammaksi. Ohjelmistoihin tulisi sisällyttää käyttäjien mahdollisuus vaikuttaa ohjelmiston energiankulutukseen. Käyttöliittymä tulisi olla sellainen, että käyttäjä pystyy selvästi havaitsemaan ohjelmiston energiankulutuksen esimerkiksi värien avulla. [61] [13]

Käyttäjälle tulisi myös tehdä helpoksi tarpeettomien energiaa kuluttavien ominaisuuksien pois kytkemisen ja yötilan käyttämisen. Myös mainosten estäminen sekä videoiden resoluution laskeminen säästää kaistanleveyttä ja energiaa. Tämä voidaan jopa automatisoida tekoälyn avulla seuraamalla toimintaa ja laskemalla käyttämättömien prosessien tilaa lepotilaan tai jopa sammuttamalla ne. [61] [49]

3.4.4 Algoritmit

Ohjelmistokehityksessä on joitain yleisiä käytäntöjä, jotka eivät ole vihreitä. Esimerkiksi kehittynyt tietokonelaitteisto ja uusin teknologia kompensoivat ohjelmistotuotteen puutteet, ja siksi hitaampia, tehottomia ja kalliimpia ohjelmistoja on ollut olemassa pitkään ilman, että suorituskyky on heikentynyt. Ohjelmistoinsinöörit tekevät kustannusanalyysin siten, että työtunteja pidetään arvokkaimpana hyödykkeenä ja siksi painopiste on kehitysajan lyhentämisessä. Tehokkaamman algoritmin kehittämistä, jonka suorittaminen vie vähemmän aikaa, ei kuitenkaan juurikaan ajatella. Koska ohjelmistoa ajetaan tuhansia ja miljoonia kertoja, ajan säästäminen suorituksessa voi olla parempi ratkaisu sekä kustannusten että kestävyyden kannalta. Ihmisinä ohjelmistosuunnittelijat vihaavat saman ongelman ratkaisemista yhä uudelleen ja uudelleen, joten rakennamme koodikirjastoja ja käytämme muita kirjastoja mahdollisimman paljon. Mutta yhä enemmän luotamme kolmannen osapuolen komponentteihin, jotka ovat tuhansia kertoja suurempia mitä todella haluamme käyttää. [3]

Koodin optimoinnilla voidaan kuitenkin saavuttaa huomattavia energiasäästöjä [1] [30]. Epäpuhtaiden funktioiden, eli funktioiden välttäminen joilla on syöteparametreja, voi vähentää energiankulutusta. Myös silmukoiden ja tietokantahakujen määrän vähentäminen uudelleen faktoroimalla voi vähentää energiankulutusta. [39]. Algoritmien energiankulutusta sekä hiilidioksidipäästöjä voidaan arvioida arviointityökaluilla, joista yksi esimerkki on `https://www.green-algorithms.org/` [30].

3.4.5 Pilvipalvelut ja reunalaskenta

Palvelujen sijoittamisella pilveen on monia energiaa säästäviä ratkaisuja ja tällä tavalla voidaan vähentää jopa 64 % energiankäytössä. Pilveen voidaan datan varastoinnin lisäksi sijoittaa laskentaa ja ohjelmiston ominaisuuksia. Pilvipohjaisissa palveluissa resursseja käytetään pyynnöstä, joka mahdollistaa isossa mittaskaalassa datakeskusten energiatehokkuuden optimoinnin. Pilvipalvelut mahdollistavat esimerkiksi hyödyllisyyslaskennan, joka on esimerkki pyynnöstä tapahtuvasta laskennasta. [61] [27]

Pilvipalveluilla on vähemmän yleiskustannuksia sekä tehokkaampi skaalautuvuus. Tässä voidaan samalla tehokkaammin hyödyntää vihreitä energianlähteitä ja esimerkiksi ottaa talteen laitteiston lämpenemisestä tuleva hukkaenergia. Tiedon käsittelemisessä keskitetyissä palvelimissa on myös se hyöty, että palvelimet ovat tehokkaammin käytössä ja niin kutsuttua joutokäyntiä on vähemmän. [61] Palvelinkeskusten energiatehokkuutta mitataan PUE-arvolla. [54]

Pilvipalvelujen käyttäminen ei kuitenkaan tarkoita automaattisesti kestävämpiä ja vihreämpiä ohjelmistoja ja ohjelmistoarkkitehtuuri voi vaatia suuriakin muutoksia. Ohjelmiston tulisi käsitellä dataa ja datan liikkumista tehokkaasti. Tällä pyritään vähentämään datan varastoinnin tarvetta, laskentaresurssien tarvetta sekä tietoliikenneresurssien tarvetta sekä reunasta pilveen, että asiakkaalle. Tämä tarkoittaa käytännössä datan siirtelyn optimoimista, kaksoisdatan poistamista sekä älykästä tietojen pakkaamista datan käyttötiheyteen perustuen. Tämän lisäksi pitäisi hyödyntää palvelitonta laskentaa sekä FaaS-toimintoa (Function-as-a-Service), jotka mahdollistavat resurssien saumattoman skaalautumisen. Datan käytön profiloinnilla ja tekoälypohjaisten profilointisovellusten avulla voidaan löytää käyttäjien toimintamalleja ja säätää sovelluksen toimintaa tarpeen mukaisesti ja lisätä tehokkuutta. [61] [31]

Palvelimien virtualisointi on hyvä tapa säästää energiaa [40] [13] [31]. Virtualisoinnin avulla yksi fyysinen palvelin pystyy vastaamaan useasta virtuaalisesta palvelimesta, jolloin palvelinta käytetään tehokkaammin ja tyhjäkäyntiä on vähemmän. Tällä tavoin voidaan säästää energiaa, mutta myös fyysisen tilan tarvetta [40] [39].

Sovelluskehittäjille vihreät palvelinkeskukset aiheuttavat haasteita, koska laitepuolen kehittyessä myös sovellusten tulisi voida hyödyntää laitatason kehitystä. Jos näin ei tapahdu, oletettu energiansäästö jää saavuttamatta. [27] Tulevaisuudessa tullaan kuitenkin hyödyntämään hajautettua tiedonhallintaa eli reunalaskentaa. Täl-

löin laskenta suoritetaan mahdollisimman lähellä ohjelmiston loppukäyttäjää, jolloin pystytään vähentämään datan siirtelystä aiheutuvaa energiankulutusta. [61]

Pilvipalvelimien käytön yleistyminen on kuitenkin aiheuttanut sen ongelman, että ne myös käyttävät entistä enemmän energiaa ja tulevaisuudessa haasteena onkin pilvipalvelimien käytön optimointi. Esimerkiksi konttiteknologiasta on saatu hyviä kokemuksia, mutta vaatii lisää tutkimusta jotta niiden potentiaalia voidaan hyödyntää mahdollisimman optimaalisella tavalla. [28] Kaikki pilvipalvelut eivät myöskään ole automaattisesti kestävän kehityksen mukaisia. Pilvipalveluja valitessa tulisi huomioida niiden ympäristöystävällisyys. Huomioitavia asioita ovat esimerkiksi energiatehokkuus sekä uusiutuvien energialähteiden käyttäminen. Pitää kuitenkin muistaa, että pilvipalvelut eivät hyödyllisyydestään huolimatta ratkaise kaikkia kestävään kehitykseen liittyviä ongelmia. [61] [2] [39] [31]

3.4.6 Ohjelmointikieli

Ohjelmointikielien energiatehokkuutta tarkastellessa ennakko-oletuksena on, että ohjelmiston energiankäyttö on sähkövirta kerrottuna ajalla. Tästä voitaisiin päätellä, että ajallisesti nopein olisi myös tehokkain. Pereira et al. [46] ovat tutkineet 27 eri ohjelmointikielen ominaisuuksia kolmesta eri näkökulmasta: suoritusaika, muistin kulutus ja energiankulutus. Tulosten perusteella he ovat vertailleet myös näiden ominaisuuksien käytön välisiä suhteita. Tuloksissa esitellään binääripuun, Fannkuch-redux-ongelman ja FASTA:n avulla muodostetut tulokset energiatehokkuuden suhteen ja myös se miten ohjelmointikieli sijoittuisi suoritusajan ja muistin käytön suhteen. Tuloksista käy ilmi, että C-pohjaiset sekä imperatiiviset kielet ovat energiankulutuksen suhteen kaikista tehokkaimpia [46] [1]. Tulokset kuitenkin jakaantuvat ratkaistavana olevan ongelman mukaisesti ja esimerkiksi kun käsitellään binääripuuta tai Fannkuch-Redux -ongelmaa, C ja C++ ovat energiatehokkaimpia. FASTA:n tapauksessa kuitenkin Rust ja Fortran olisivat energiatehokkaimpia. Normalisoiduissa tuloksissa C, Rust ja C++ ovat kolme ensimmäistä energiatehokkainta ohjelmointikieltä. Tulosten perusteella käy ilmi, että kulutettu energia ei ole suoraan verrannollinen kulutettuun aikaan ja tulokset ovat riippuvaisia kielistä ja käsiteltävänä olevasta ongelmasta. Pereira et al. [47] ovat myöhemmässä tutkimuksessaan tulleet samaan johtopäätökseen. Ohjelmointikielen energiankulutus on vain joissain tapauksissa riippuvainen suoritusajasta. Vaikka tuloksista käy ilmi, että energiatehokkain kieli on usein myös nopein, ei ole yhtä ainoaa, joka on aina parempi kuin jokin toinen. Kielen energiatehokkuus on riippuvainen ohjelmointiongelmasta, jo-

hon sitä aiotaan käyttää. Esimerkiksi regex-redux vertailussa tulkitut kielet vaikuttavat olevan energiatehokkaampi valinta, vaikka tulkitut kielet eivät ole kovinkaan energiatehokkaita muita skenaarioita tarkasteltaessa.

Artikkelien [46] [47] kirjoittajat ovat tutkineet myös sitä voiko ohjelmointikielen valintaa automatisoida optimointitavoitteiden avulla. He käyttivät Pareto-optimointia, koska on mahdollista, että ei ole vain yhtä ainoaa vaihtoehtoa optimaalisimmaksi kieleksi ohjelmointiongelmaan vaan joukko kieliä, jotka ovat saman arvoisia ohjelmointiongelman ratkaisemissa. Tätä kutsutaan Pareto-optimaaliseksi joukoksi. Esimerkiksi ajan- ja muistinkäytön suhteen C, Pascal ja Go ovat saman arvoisia sekä energian- ja muistinkäytön suhteen C ja Pascal ovat saman arvoisia. Tuloksista käy ilmi, että energian- ja ajankäytön optimoimisen suhteen parhaimman ohjelmointikielen valitseminen on mahdollista ja Pareto-optimaalisia kielijoukkoja oli vain muutama.

Kääntämisen aikainen ohjelmiston optimointi voi vähentää ohjelmiston energiankulutusta. Energiankulutuksen väheneminen johtuu kuitenkin pääasiassa ohjelmiston ajamiseen käytettävän ajan vähenemisestä, joka vähentää myös ohjelmiston energiankulutusta. [25] [41]

3.4.7 Datan ja muistin käyttö

Pereira et al. [46] ovat vertailleet tutkimuksessaan myös ohjelmistokielien vaikutusta huippumuistin (peak memory) energiankäyttöön. Heidän tulostensa mukaan DRAM-muistin energiankulutus korreloituu hyvin vähän muistin käytön suhteen. Keskimäärin käännetyt kielet kuluttivat DRAM:n suhteen 14J, virtuaalikoneen kielet 52J ja tulkitut kielet 236J. Myöhemmässä tutkimuksessaan Pereira et al. [47] tulivat siihen tulokseen, että koodin optimointi, jonka tarkoituksena on vähentää prosessorin energiankulutusta voi vähentää myös DRAM-muistin energiankulutusta. Viisi parasta kieltä DRAM-muistin energiankulutuksen suhteen olivat C, Rust, C++, Ada ja Java. He havaitsivat myös, että kokonaismuistin määrän ja DRAM-muistin energiankulutuksen välillä on suora yhteys. Mitä enemmän muistia ohjelmisto käyttää elinkaarensa aikana, sitä enemmän kuluu energiaa DRAM-muistin käyttämiseen.

Moises et al. [39] tutkimuksessa kerrotaan, että sivujen välimuisti voisi vähentää verkkosivujen energiankulutusta. Kuitenkin Malavolta et al. [35] uudemmassa tutkimuksessa on tultu siihen tulokseen, että välimuistin käytöllä ei ole vaikutusta verkkosivun energiankulutukseen merkittävissä määrin.

Taulukko 3.1: Ohjelmointikielien vertailun normalisoidut tulokset energian, ajan ja muistin suhteen. Mukaillen [46].

Kieli	Energia
C	1.0
Rust	1.03
C++	1.34
Ada	1.70
Java	1.98
Pascal	2.14
Chapel	2.18
Lisp	2.27
Ocaml	2.40
Fortran	2.52
Swift	2.79
Haskell	3.10
C#	3.14
Go	3.23
Dart	3.83
F#	4.13
JavaScript	4.45
Racket	7.91
TypeScript	21.50
Hack	24.02
PHP	29.30
Erlang	42.23
Lua	45.98
Jruby	46.54
Ruby	69.91
Python	75.88
Perl	79.58

Kieli	Aika
C	1.00
Rust	1.04
C++	1.56
Ada	1.85
Java	1.89
Chapel	2.14
Go	2.83
Pascal	3.02
Ocaml	3.09
C#	3.14
Lisp	3.40
Haskell	3.55
Swift	4.20
Fortran	4.20
F#	6.30
JavaScript	6.52
Dart	6.67
Racket	11.27
Hack	26.99
PHP	27.64
Erlang	36.71
Jruby	43.44
TypeScript	46.20
Ruby	59.34
Perl	65.79
Python	71.90
Lua	82.91

Kieli	Mb
Pascal	1.00
Go	1.05
C	1.17
Fortran	1.24
C++	1.34
Ada	1.47
Rust	1.54
Lisp	1.92
Haskell	2.45
PHP	2.57
Swift	2.71
Python	2.80
Ocaml	2.82
C#	2.85
Hack	3.34
Racket	3.52
Ruby	3.97
Chapel	4.00
F#	4.25
JavaScript	4.59
TypeScript	4.69
Java	6.01
Perl	6.62
Lua	6.72
Erlang	7.20
Dart	8.64
Jruby	19.84

Taulukko 3.2: Pareto-optimaaliset kielijoukot. Mukaillen [46].

Aika ja muisti	Energia ja aika	Energia ja muisti	Energia, aika ja muisti
C - Pascal - Go	C	C -Pascal	C - Pascal - Go
Rust - C++ - Fortran	Rust	Rust - C++ - Fortran - Go	Rust - C++ - Fortran
Ada	C++	Ada	Ada
Java - Chapel - Lisp - Ocaml	Ada	Java - Chapel - Lisp	Java - Chapel - Lisp - Ocaml
Haskell - C#	Java	Ocaml - Swift - Haskell	Swift - Haskell - C#
Swift - PHP	Pascal - Chapel	C# - PHP	Dart - F# - Racket - Hack - PHP
F# - Racket - Hack - Python	Lisp - Ocaml - Go	Dart - F# - Racket - Hack - Python	JavaScript - Ruby - Python
JacaScript - Ruby	Fortran - Haskell - C#	JavaScript - Ruby	TypeScript - Erlang
Dart - TypeScript - Erlang	Swift	TypeScript	Lua - JRuby - Perl
JRuby - Perl	Dart - F#	Erlang - Lua - Perl	
Lua	JavaScript	JRuby	
	Racket		
	TypeScript - Hack		
	PHP		
	Erlang		
	Lua - JRuby		
	Ruby		

3.4.8 Sähköinen jäte

Sähköinen jäte voidaan ymmärtää joko elektroniikkajätteenä tai datajätteenä, joka kuluttaa tietokone- ja henkilöstöresursseja. Tässä luvussa sähköinen jäte tarkoittaa datajätettä.

Sähköinen jäte lisää energiankäyttöä sekä hiilijalanjälkeä ja on siksi kriittinen menestystekijä vihreässä ohjelmoinnissa [6] [51]. Sähköistä jätettä syntyy vaatimusmäärittelyvaiheessa esimerkiksi silloin, kun tuotetta ei välttämättä oteta käyttöön ollenkaan, kun virheellisten vaatimusten takia koodia joudutaan kirjoittamaan uudestaan vastaamaan asiakkaan vaatimuksia, koodi tai dokumentaatio on liian monimutkaista tai yksityiskohtaista, ohjelmisto aiheuttaa ylimääräistä kuormitusta, odottaminen, viivästyminen sekä puutteellinen kommunikaatio. [6]

Implementointivaiheessa sähköisen jätteen määrää lisää esimerkiksi koodin uudelleen kirjoittaminen sekä ohjelmiston toteuttamisesta aiheuttama ylimääräinen kuormitus. Tämä tarkoittaa ylimääräisten sekä virheellisten ominaisuuksien toteuttamista ohjelmistoon. [6] Asiakkaalle tulisi toimittaa siis ohjelmisto, jossa on vain välttämättömät, oikeasti käyttöön tulevat ominaisuudet. [13] Myös ohjelmoijien puutteelliset taidot saattavat lisätä sähköisen jätteen määrää. Parhaiten tätä voidaan välttää toimimalla sovittujen toimintamallien mukaan alusta lähtien. Sähköisen jätteeseen liittyvät ongelmat tulisi ratkaista pikaisesti ongelman havaitsemisen jälkeen.

Testausvaiheessa käyttäjätestauksen palautteen odotteleminen luo sähköistä jätettä. [6] Sähköisen jätteen määrää voidaan vähentää esimerkiksi pilvipalvelujen käytöllä. [2]

4 Kyselyn rakentamisesta

Informatiivisia kyselyitä voidaan hyödyntää opetuksen apuna. Kyselyillä voidaan arvioida vastaajan tietoja ja ajatuksia. Informatiivisen kyselyn tuottamiseen käytetty aika tuottaa paljon hyötyä tehostetussa oppimisessa, joka saavutetaan niiden käytöllä. Informatiivisten kyselyiden tuottaminen on hyödyllistä siinäkin mielessä, että ne voidaan rakentaa usein sillä materiaalilla, joka on jo suunniteltua kurssimateriaalia ja ne validoivat opiskelijan oppimista paremmin kuin tavalliset opiskelijaarviointimenetelmät. [42]

Optimaalisen kyselyn suunnitteluun on olemassa joukko ohjeita. Kyselyä rakentaessa tulisi käyttää yksinkertaisia ja tuttuja sanoja. Moniselitteisiä sanoja ei pitäisi käyttää, jotta kaikki vastaajat pystyisivät tulkitsemaan kysymykset samalla tavalla. Murre- ja ammattisanastoa tulisia välttää. Kysymyksenasettelun syntaksi tulisi olla yksinkertaista ja sanamuotojen tulisi olla täsmällisiä. Kysymyksenasettelut ei saisi olla johdattelevia ja kysymysten vastausvaihtoehdot tulee olla toisensa poissulkevia. Kysymyksessä tulisi kysyä vain yhdestä asiasta kerrallaan ja useammasta asiasta kysymistä tulisi välttää. Kysymyksissä tulisi välttää myös negaatioita. Kyselyn ensimmäisiin kysymyksiin tulisi olla helppoja vastattavia. Samaa aihetta koskevat kysymykset tulisi ryhmitellä yhteen ja samaa aihetta koskevat kysymykset tulisi edetä yleisestä aiheesta erityisiin kysymyksiin. Arkaluontoiset kysymykset olisi hyvä sijoittaa kysymyksen loppuun. Kyselyille tulisi suorittaa myös esitestaus. [29]

Hannulan ja Lönnqvistin [20, s. 46–56] mukaan mittarilla tarkoitetaan täsmällisesti määriteltyä menetelmää, jonka avulla pyritään selvittämään jonkin tietyn menestystekijän suorituskykyä. Menestystekijä on strategisen toteutumisen kannalta keskeinen asia ja jokaisella yrityksellä ja sen henkilöstöllä on omat menestystekijät, jotka ovat heidän toimintansa kannalta oleellisia.

Henkilöstön osaamisen arvioiminen tuottaa yritykselle hyödyllistä tietoa esimerkiksi osaamisen hyödyntämisen, kehittämistoimenpiteiden suunnittelun sekä henkilöstön oppimismotivaation virittämisen kannalta. Yrityksen johto pystyy tulosten perusteella kohdistamaan resursseja sellaisiin alueisiin, joissa tarvitaan vahvistamista. [62, s. 153]

Mittareiden kehittäminen on usein iteratiivinen prosessi. Mittarin hyvyyttä on

tarkoituksenmukaista arvioida prosessin eri vaiheissa. Tarvittaessa mittaria täytyy muokata, jos se ei vastaa sitä tarkoitusta ja tavoitteita, joihin mittaria aiotaan käyttää. Mittaria tulisi tarkastella suhteessa liiketoiminnan tavoitteisiin ja tämän tulisi vastata tavoitteita erityisesti silloin, kun mittarin tarkoituksena on ohjata toimintaa tavoitteiden saavuttamiseksi. Mittarin toimivuutta voidaan arvioida konkreettisten kokemusten avulla ja tämä voidaan viemällä mittari oikeaan käyttöympäristöön. Oikeiden kokemusten perusteella mittaria voidaan kehittää edelleen paremmaksi. Toisaalta joskus mittarit saattavat osoittautua merkityksettömiksi ja liiketoiminnan kannalta ne on järkevää poistaa käytöstä. [34, s. 189] Ennen mittarin suunnittelemista tulisi olla selvillä, miksi menestystekijöitä tai osaamista mitataan. Uusi-Rauvan [s. 11][59] mukaan mittareita voidaan tehdä esimerkiksi oppimis- ja informaatiotarkoituksiin.

Verkossa tapahtuva opiskelu on yleistynyt paljon viimeisen vuosikymmenen aikana ja sillä voi olla positiivisia vaikutuksia oppimiseen. Verkko-oppimisella voidaan saavuttaa konstruktiivisia oppimistuloksia, koska verkon avulla tapahtuva opiskelu mahdollistaa opiskelun milloin ja missä tahansa [32] [9]. Encarnacion et al. [16] tutkimuksessa verkko-opiskelua opiskeluun ja opettamiseen hyödyntävät henkilöt pitävät verkkossa tapahtuvaa opiskelua tehokkaana tapana opiskella. Tutkimuksessa opiskelijat kokivat verkko-opiskelulla olevan positiivisia vaikutuksia heidän oppimistuloksiin. [16]

Tässä Pro Gradu -tutkielmassa tarkoituksena on rakentaa informatiivinen kysely, jolla pyritään samalla kartoittamaan henkilöstön vihreän ohjelmointiosaamisen tasoa ja selvittää näin mahdollisia kehittämiskohteita. Kyselyn tarkoituksena on myös jakaa tietoa vihreästä ohjelmoinnista ja antaa ideoita kyselylomakkeen läpikäyneille, joilla voitaisiin mahdollisesti parantaa ohjelmistojen ja ohjelmistokehitysprosessin vihreyttä. Sähköinen kysely mahdollistaa toiminnallisuuksia, joita tavallisessa kyselyssä ei ole mahdollista. Sähköisseä kyselyssä voidaan esimerkiksi antaa välitöntä palautetta ja informaatiota. Tämän lisäksi sähköisen kyselyn voi tehdä silloin kun se työntekijälle parhaiten sopii.

5 Ensimmäinen iteraatio

Tässä luvussa esitetään ensimmäisen iteraation kulku. Alustava versio kyselystä tehtiin kirjallisuuskatsaukseen pohjautuen. Kysely löytyy liitteestä B. Ennen teemahaastattelujen aloittamista kysely käytiin läpi toimeksiantajan yhteyshenkilön kanssa. Tällöin tarkistettiin, että kysely vastaa toimeksiantajan ajatusta artefaktista ja että kysely sisältää tarpeelliset aihealueet.

5.1 Ensimmäisen iteraation kulku

Ensimmäisen iteraation tavoitteena oli selvittää, onko kysely tehokas metodi kartoittaa ja lisätä ohjelmistokehittäjien vihreää ohjelmointiosaamista sekä selvittää motivoiko kysely lisäämään vihreitä ohjelmointikäytänteitä käytännön työhön. Näiden lisäksi tavoitteena oli kerätä palautetta kyselyn alustavasta versiosta ja hyödyntää tätä palautetta kyselyn seuraavaa versiota kehitettäessä.

Tulosten perusteella kyselyn ensimmäisessä versiossa aihealueita tulisi muokata siten, että ohjelmistokehitysprosessin osuudesta poistettiin ketterään kehitykseen / Scrum-projektinhallintaan liittyvät kysymykset. Tämän takia näitä kysymyksiä koskevia parannusehdotuksia ei myöskään tarvitse ottaa huomioon jatkokehityksessä. Näiden kysymysten lisäksi tulisi poistaa yksittäisiä kysymyksiä kyselyn muista osioista. Elinkaarimallia koskevat kysymykset koettiin irrallisina, joten tulisi muokata yhtenäisempiä. Pariohjelmointia koskeva osio tulisi siirtää elinkaarimallin suunnitteluvaiheosioon.

Kysymyksenasettelut koettiin liian jyrkkinä, ohjailevina sekä epäselvinä, joten näitä tulisi pyrkiä muuttamaan mahdollisuuksien mukaan kevyempään suuntaan, vähemmän ohjaileviksi sekä selventää tarvittavilta osin. Polaarisiin kysymyksiin lisätään vastausvaihtoehto "En tiedä"ja vastaamisen jälkeen tullaan tarjoamaan vastaajalle oikea vastausvaihto. Kysymysten yhteyteen lisätään termistöä koskevia selityksiä. Tämän lisäksi suomenkielisten termien lisäksi tullaan esittämään myös termin englanninkielinen versio. Kyselyn alkuun lisätään kyselyn sisällöstä kertova hakemisto sekä alustus kyselyn tarkoituksesta ja tavoitteista.

Kaikki vastaajat pitivät sähköistä kyselyä hyvänä ideana, joten kysely siirretään

sähköiseen muotoon. Sähköinen versio mahdollistaa myös ominaisuuksia, joita vastaajat toivoivat kyselyssä olevan. Näitä ovat esimerkiksi kyselyn etenemisestä kertova graafinen indikaattori sekä oikean vastausvaihtoehdon kertominen heti vastaamisen jälkeen.

Kyselyn sähköinen versio tullaan toteuttamaan LimeSurvey-palvelussa ja se löytyy osoitteesta: `https://cinetcampus.fi/survey21/index.php/497553?newtest=Y&lang=en`

5.2 Ensimmäisen teemahaastattelun runko

Tässä osassa esitetään teemahaastattelujen runko. Teemat rakentuivat haastateltavien taustatietojen, kyselyn sisällön, kyselyn formaatin ja kyselyn ulkoasun ympärille. Teemojen alla olevat tarkentavat kysymykset toimivat apuna haastattelijalle haastattelun aikana.

1. Haastateltavan taustatiedot:
   (a) Nimi
   (b) Ammatti
   (c) Työtehtävät
   (d) Työkokemus
   (e) Onko ohjelmointiin tai ohjelmistokehitysprosessiin liittyvät käytänteet entuudestaan tuttuja?
   (f) Kokeeko haastateltava vihreään ohjelmointiin liittyvän osaamistason hyväksi vai kaipaisiko hän lisää koulutusta asiaan liittyen?
   (g) Mikä on haastateltavan suhtautuminen vihreyteen?
   (h) Miten vihreiden käytänteiden toteutuminen näkyy omassa työssä / tiimissä / organisaatiossa?
   (i) Onko haastateltava tutustunut etukäteen aiheena olevaan kyselylomakkeeseen?
2. Kyselylomakkeen sisältö
   (a) Mitä mieltä haastateltava on kyselylomakkeesta?
      i. Onko kyselylomake hyödyllinen?

(b) Mitä mieltä haastateltava on kysymyksistä?

i. Onko kysymykset liian helppoja tai vaikeita?

ii. Onko kysymykset informatiivisia?

iii. Onko väittämissä "kyllä/ei-vastausvaihtoehto riittävä vai olisiko hyödyllistä lisätä "En tiedä-vastausvaihtoehto?

(c) Mitä mieltä haastateltava on kyselylomakkeessa olevista aiheista?

i. Oliko kaikki aiheet tarpeellisia?

ii. Mikä aihe nousi esille hyödyllisenä?

iii. Onko kaikki aiheet tarpeellisia ohjelmointityötä ajatellen?

iv. Olisiko lomake kaivannut lisää jostain aiheesta tai puuttuiko jokin aihe kokonaan?

(d) Mitä mieltä haastateltava on vihreää ohjelmistokehitysprosessia käsittelevistä kysymyksistä ja aiheista?

(e) Mitä mieltä haastateltava on vihreää elinkaarimallia käsittelevistä kysymyksistä ja aiheista?

(f) Mitä mieltä haastateltava on vihreää ohjelmistoa käsittelevistä kysymyksistä ja aiheista?

3. Kysymysten formaatti

(a) Onko kysymystenasettelu selkeä?

(b) Minkälaiset kysymykset haastateltava kokee hyvänä?

(c) Onko kieliasu ymmärrettävää?

4. Kyselylomakkeen ulkoasu

(a) Mitä mieltä haastateltava on kyselylomakkeen ulkoasusta?

i. Onko runko selkeä?

ii. Erottuvatko aihekokonaisuudet hyvin toisistaan?

iii. Eteneekö lomake loogisesti?

iv. Onko kyselylomake sopivan pituinen?

v. Olisiko hyödyllistä jakaa kyselylomake kahteen tai useampaan osa-alueeseen ja vastattavat aihealueet valikoituisivat työntekijän työnkuvan perusteella?

5.3 Ensimmäisen iteraation haastattelujen tulokset

Ensimmäisen haastattelun tuloksissa käsitellään kyselylomakkeen informatiivisuuteen ja hyödyllisyyteen, aihealueisiin, kysymyksiin, ulkoasuun sekä kieliasuun, termistöön sekä sähköisen kyselylomakkeen toteuttamiseen liittyvät tulokset.

5.3.1 Kyselyn informatiivisuus ja hyödyllisyys

Haastattelun aikana kysyttiin haastateltavien mielipidettä liittyen kyselylomakkeen hyödyllisyyteen ja informatiivisuuteen. Haastateltavat olivat sitä mieltä, että kyselylomake on informatiivinen ja ajatuksia herättävä. He kertoivat myös, että kyselylomakkeesta jää mieleen asioita ja se parantaa heidän mielestään ohjelmistokehityksen laatua.

"Ihan siis kattavasti tai siis opin ihan tämän kyselylomakkeen perusteellakin uusia asioita varmaankin, että ihan mielenkiintoinen ja kattavan oloinen."

"Tää parantaa myös sitä ohjelmistokehityksen laatua sen lopputuloksen laatua, vaikka se pitäisi tulla jo ilman tätäkin."

Haastateltavat pitivät kyselylomaketta pääasiassa hyödyllisenä tai osittain hyödyllisenä. Yksi haastateltavista oli sitä mieltä, että kyselylomake ei ole hyödyllinen.

"Joo kyllä mä sanoisin, että tää on hyödyllinen just ehkä siitä näkökulmasta, että sellainen tyyppi joka ei ole ihan hirveän paljon miettinyt näitä asioita niin tavallaan tästä niin kun käy aika hyvin ilmi ainakin toi asian kokonaisvaltaisuus..."

"Että varmasti uutta tietoa olisi tässä ainakin muutamat termit mitkä on nyt voisi tästä googlettaa, mutta ehkä niitä lukuun ottamatta en nyt nähnyt tätä itselleni silleen hyödyllisenä."

5.3.2 Kyselyn aihealueet

Haastateltavat pitivät elinkaarimallia, vihreää ohjelmistoa, ohjelmistokehitysprosessia, dokumentaatioon ja viestintään sekä tehokkuuteen käsittelevien aihealueiden kysymyksiä hyvinä. Työtapoja koskevat kysymykset olivat helposti lähestyttäviä ja

nopeita vastattavia sekä testaus- ja ylläpitovaiheen kysymykset helppoja vastattavia. Mittaamiseen liittyvät kysymykset koettiin hankalina vastattavina.

"Täälläkin (vihreää ohjelmistoa käsittelevissä kysymyksissä) on tosiaan ihan mielenkiintoisia kysymyksiä."

"Sitten tuli näitä vähän hankalampia taas, tähän vihreään ohjelmistoon liittyvät kysymykset ohjelmiston vihreiden mittaaminen..."

Haastateltavien kokemukset aiheiden tarpeellisuudesta sekä kattavuudesta vaihtelivat. Haastateltavien mielestä kaikki kyselylomakkeessa olevat aiheet olivat tarpeellisia ja mitään aihetta ei ole tarpeellista poistaa kyselylomakkeesta.

"Kyllä juuri näin ja sitten se kokonaisuus ratkaisee, se on hyvä että tää käsittelee näitä asioita niin monesta eri näkökulmasta."

Toisaalta erityisen hyödyllistä aihetta kyselylomakkeen kannalta ei ollut.

"...en mä ehkä osaa sanoa, että mikä täällä olisi erityisen hyödyllinen..."

Ohjelmistokehitysprosessia ja ohjelmiston elinkaarimallia vertailtaessa koettiin ne yhtä tärkeinä tai ohjelmiston elinkaarimalli koettiin parempana. Kaikista aihealueista koettiin olevan kattavasti kysymyksiä. Yksi vastaajista olisi kuitenkin toivonut joihinkin aiheisiin lisätietoa.

"Tietyssä mielessä mä ehkä nää elinkaarikysymykset näen silleen geneerisempinä, joka tietyllä tapaa on ehkä ihan hyvä asia, koska geneerisemmät on ehkä sovellettavissa helpommin."

"Mutta siis kyllähän se monipuolinen oli ja monelta näkökulmalta sitä asiaa pyörittää, että siinä mielessä ihan kattava mun mielestä."

"Pitäisi palata takaisin niihin kysymyksiin, mutta ehkä sellainen fiilis mulla tällä hetkellä, että joistakin asioista tai aiheista olisi voinut olla lisääkin tietoa."

Todellisten käytäntöjen kannalta kehittämisvaihetta ja ohjelmistokehitystä koskevat kysymykset olivat oleellisimpia.

"No siis varmaan ehkä tottakai itselleni toi kehittämisvaihe tietysti on aina se niin kun kiinnostavin, se on sitä omaa tekemistä eniten."

"...niin tavallaan on asioita, joihin mihin me Digiana IT toimittajana, just tässä asiakkuudessa voidaan vaikuttaa, just siihen ohjelmistokehitykseen."

Sen sijaan ohjelmistokehitysprosessiin, pariohjelmointiin, tarkistuslistoihin ja testausvaiheeseen liittyvät kysymykset eivät vastanneet oikeita käytänteitä.

"Tässä on, esimerkiksi kehittämisvaiheessa, on pariohjelmoinnista kysymys, mä veikkaan, että sitä pariohjelmointia ei välttämättä hirveästi käytetä meillä..."

"Ja tuota ehkä sitten toi testaus oli ehkä semmoinen mikä ei sitten itselle ole niin ehkä lähellä."

Haastateltavat toivat ilmi joitain kyselylomakkeessa olevia kehitettäviä asioita. Ohjelmistokehitysprosessiin, prosessin mittaamiseen ja elinkaarimallin osuus koettiin raskaana. Ohjelmistokehitysprosessin osuudessa tulisi myös käyttää geneerisempää ilmaisua "iteraatio"sanan "sprintti"sijasta.

Sitten toi ohjelmistokehitysprosessiin liittyvät kysymykset, prosessin mittaaminen ja elinkaarimalli niin mä veikkaan, että näissä on aika monta eri vaihtoehtoa, nää voi hidastaa varsinkin tässä kehittämisvaiheessa, niin nää on vähän semmoisia, että näissä varmaan voi energiaa alkaa näitten jälkeen loppumaan.

"Siinä mielessä, että ne oli ihan tässä kysymyksessäkin sprinttiin/iteraation, niin voi tietysti ajatella, että iteraatio ei välttämättä aina tarkoita tiettyä yksittäistä sprinttiä vaan jotain iteraatiota."

Elinkaarimallia koskevat kysymykset koettiin irrallisina ja pariohjelmointia koskeva kysymys tulisi sijoittaa uudelleen suunnitteluvaiheosioon.

"...itselleni nää yksittäiset kysymykset vähän tuntuu irrallisilta (elinkaarimallia käsittelevässä osiossa)."

"Tavallaan se (pariohjelmointi) varmaan kuuluisi olla suunnitteluvaiheessa, että tavallaan kun lähdetään sitä ohjelmistoa arkkitehtuuria suunnittelemaan, että otetaanko siinä huomioon."

5.3.3 Kyselyn kysymykset

Haastateltavat kokivat kysymykset pääasiassa vaikeina.

"Nää oli vähän vaikeita kysymyksiä."

Toisaalta haastateltavat kokivat, että kysymykset eivät olleet helppoja eikä vaikeita tai kysymykset olivat helppoja ja vaikeita.

"...en oikeastaan osaa nähdä sitä niin, että helppoja tai vaikeita."

"... osa oli siis semmoisia, että helpommin saa vastattua ja osa oli sitten, että vähän vaikeampi vastata..."

Kaksi haastateltavaa piti kysymyksiä helppoina tai osittain helppoina. Kysymysten vaikeus koettiin riippuvan vastaajan taustasta ja työtehtävistä. Vastauksissa korostui se, että johtotehtävissä työskenteleville ja ohjelmointia osaamattomille henkilöille kysymykset ovat vaikeampia.

"... niin sitten mä ajattelin, että jos tavallaan kyselylomakkeeseen vastaa semmoinen management tason henkilö, joka ei ole kauheasti enää kehittäjä, niin nää voi olla vaikeita."

Haastateltavat kokivat kysymyksenasettelun jyrkäksi ja ohjailevaksi.

"Ja ehkä siitä kyselylomakkeesta, niin tuli semmoinen kovin mustavalkoinen olo sen kanssa, että moneen kysymykseen, että jos kysytään vaan että vastaat a tai b, niin tuli vähän semmoinen fiilis kun aika monestikin tämmöisissä monivalintakysymyksissä, että eikö nyt oikeasti ole mitään välimaastoa, että aika osa oli aika semmoisia omasta näkökulmasta semmoisia äärimmäisyyksiä."

"...että nää kysymykset on vähän ehkä ohjailevia, että mä veikkaan, että sillä ei tietämättä mitään aiheesta tästä tulisi ihan kohtalaisen hyvät pisteet."

He myös kokivat, että kysymyksenasettelu on osittain epäselvä ja osittain selkeä.

"Niinku sanoin, että tuskailin vähän sitä, että mitä tässä pitää vastata, että just se itsessään vai suhteessa johonkin, se tuntui itselleen vähän epäselvältä ja sitten mikä on, puhutaan merkittävästi, niin mikä itseasiassa on merkittävä, et onko mun merkittävä nyt sama kuin kysymyksenasettelijan merkittävä?"

Polaariset kysymykset koettiin sopivan joihinkin aiheisiin ja silloin kun kysytään kysymyksiä, joihin on vain yksi oikea vastaus.

"Kyllä ne tietyssä asiassa toimii tämmöiset kyllä ei kysymykset."

Näiden koettiin olevan herätteleviä sekä nopeampia vastata. Osa polaarisista kysymyksistä oli haastavia ja osa oli helppoja vastattavia. Näiden kysymysten koettiin olevan myös johdattelevia.

"Siis siinä mielessä, että tavallaan näihin on se teknisesti, oli se sitten kyllä/ei tai tommoinen useampi monivalinta, näihin on tosi nopea sinänsä vastata."

Ne kyllä/ei kysymykset on ehkä just semmoisia mistä mulla tulee tää ns. johdatteleva fiilis tai semmoinen.

Monivalintakysymyksiä pidettiin loogisina ja helppoina vastata, mutta haastavina.

"No siis joo ihan niin kuin kaikki tämmöiset kyselyt missä on monivalintoja (on loogisia ja helppoja vastata)."

"...useimmiten noi missä on sitten enemmän näitä sanallisia vaihtoehtoja, niin sitten tietysti niissä joutuu pikkaisen enemmän miettimään itsekin, että mitenköhän se nyt sitten menee."

Näissä myös ajateltiin olevan hyvä ohjeistus, mutta monivalintavaihtoehtoja sisältävät kysymykset eivät sovellu hyvin kaikkiin kysymyksiin. Monivalintakysymyksissä olevat käänteiset vastausvaihtoehdot eivät olleet loogisia.

"Mutta ei varmaan kaikkia kysymyksiä ei pysty muotoilemaan monivalinnoiksi."

"Sitten täällä oli näitä valintakohtia niin niissähän tai aika monessa valintakohdassa on silleen, että on niin kun mun käänteiset valinnat, että tavallaan siinä siis se ei niin kun äkkiseltään tunnu ihan loogiselta, että sulla on niin kun eri valintoja, sitten jokaisesta niistä on vielä käänteinen valinta olemassa."

Haastateltavat pitivät siitä, että kyselylomakkeessa oli sekä polaarisia ja monivalintakysymyksiä. Kuitenkin monivalintakysymyksiä tulisi suosia polaaristen kysymysten sijaan. Se, että vapaan tekstin kenttiä ei ole, oli hyvä asia.

"Mutta kyllä mä itse tykkäsin just tästä, että tässä on välillä näitä kyllä/eitä ja välillä on sitten näitä tämmöisiä monivalintoja."

"Mutta lähtökohtaisesti semmoiset mitkä pystyy muotoilemaan monivalinnoiksi niin."

Osa haastateltavista oli sitä mieltä, että "en tiedä-vastausvaihtoehto tulisi lisätä vastausvaihtoehdoksi kysymyksiin ja osa oli sitä mieltä, että sitä ei tulisi lisätä vastausvaihtoehdoksi.

"Jos tosiaan tää on enemmän just semmoinen niin kun tietoutta lisäävä niin sitten ehkä se "en tiedä"olisi sinällään ihan hyvä, että jos ei ole tarkoituskaan pitää tätä semmoisena tenttimäisenä, että pitäisi saada ne vastaukset oikein vaan ehkä enemmän just semmoista tietoisuuden tasoa mittavana kyselynä, niin sitten se "en tiedä"olisi silleen hyvä."

"Että tosiaan se en osaa sanoa tulee turhan heppoisin perustein sitten, että en mä nyt oikein tiedä tästä aiheesta niin laitan "en osaa sanoa"."

Kyselylomakkeessa tulisi vastaamisen jälkeen tarjota oikea vastausvaihtoehto vastaamisen jälkeen.

"Musta se on hyvä kognitiivisestikin hyvä sellainen, että sä otat siitä b ja no väärin meni koska a ja c olisi ollut näin, se on musta hyvä."

Haastateltavat kertoivat pitävänsä joitain kysymyksiä epäoleellisina, itsestään selvinä tai näiden katsottiin olevan kyseenalaisia hyödyllisyyden kannalta.

"Tää (Onko totta, että kestävä kehitys voidaan määritellä tarkoitukseen sopivuudella, reduktiolla ja kauneudella?) on todella hassu silleen, että en nyt silleen ihan suoraan tästä esimerkiksi ymmärrä mitä tässä nyt niin kun tarkoitetaan."

"...mä en osaa sanoa onko esim. just tossa, vähän tuollaista niinku agilekulmaan, että onko siitä aidosti tuossa vihreässä ohjelmistokehityksessä hyötyä vai onko siitä hyötyä noin ylipäänsä ohjelmistokehityksessä."

Kyselylomakkeeseen tulisi lisätä kysymyksiä liittyen ohjelmiston vihreyden mittaamiseen, hukkaenergian vähentämiseen ja optimointiin, sisäisen kehitysprosessin tehostamiseen, skaalautuvuuteen, itse ohjelmointiin, itse vihreästä ohjelmistokehitykseen sekä pilvipalveluihin. Näiden lisäksi elinkaarimallin ylläpito- ja suunnitteluvaiheen osuuteen tulisi lisätä kysymyksiä.

"Tässä on tosi paljon muitakin hyviä tulokulmia, mutta jos nyt yksi pitää valita, niin mä nostan tuon (vihreä ohjelmisto ja vihreyden mittaaminen) sen takia kärkeen, että se on varsinkin teknisille ihmisille helposti lähestyttävin, semmoinen millä pystytään parantamaan asioita mittaamalla niitä."

"Ja mun mielestä se kantava teema, mikä tuossa ehkä kuitenkin ehkä tuli sieltä rivien välistä, on nimenomaan se hukkaenergia ja sen tyyppinen optimointi minkä ehkä tiesin jo vähän ennakkoonkin jo, mutta se on mun mielestä ihan semmoinen asia mitä on myös syytä korostaa."

5.3.4 Kyselyn ulkoasu

Haastateltavat kokivat kyselylomakkeen etenevän loogisesti ja kyselylomake olevan rakenteeltaan selkeä. Yksi vastaajista ei osannut ottaa kyselylomakkeen loogisuuteen kantaa. Työtapoja koskeva osuus koettiin kuitenkin irralliseksi.

"Joo kyllä tää ihan loogisesti mun mielestä etenee."

"...se, että eteneekö se jotenkin loogisesti en osaa siihen vastata"

Osa haastateltavista piti kyselylomaketta liian pitkänä ja sen pituutta tulisi heidän mukaansa vähentää. Toisaalta kyselylomake koettiin olevan sopivan pituinen.

"Aika laaja se oli tietysti, että siinä oli paljon kysymyksiä."

"Että varmaan siinä käytännön kyselylomakkeessa, niin voisi olla tiivistettävää"

Osa haastateltavista oli sitä mieltä, että kyselylomake voitaisiin jakaa useampaan osa-alueeseen, joihin vastaaminen perustuisi työntekijärooliin. Osa-alueet voisivat jakautua kaikille yhteisiin kysymyksiin, ohjelmistokehittäjille suunnattuihin kysymyksiin sekä ohjelmistokehitysprosessiin liittyviin kysymyksiin.

"Tai siis olisi tavallaan esimerkiksi kaikille yhteisiä kysymyksiä ja sitten voisi olla ehkä roolipohjainen jonkinlainen jaottelu tai vaihtoehtoisia."

"...mä vähän luulen, että riippuu vähän vastaajan esimerkiksi teknisestä tasosta (pitäisikö kyselylomake jakaa erillisiin osa-alueisiin)"

Toisaalta kysymykset voisi jakaa ohjelmistokehittäjille suunnattuihin kysymyksiin sekä syventäviin kysymyksiin. Työtapoihin liittyvät kysymykset voisivat olla kaikille yhteisiä ja elinkaarimallia koskevat kysymykset voisi osoittaa ohjelmistokehittäjille.

"Voisi olla, mutta ei ehkä välttämättä enempää tarvitsisi olla kuin kaksi, että tavallaan olisi se itse ohjelmistokehitys niin, jos siihen löytää syventäviä kysymyksiä, niin sitten voisi sitä laajentaa..."

"Mun mielestä kaikki nää ylhäällä olevat asiat on yhteisiä ja sitten voi voisi ehkä miettiä vihreään ohjelmistokehitysprosessiin liittyvät."

Elinkaarimallia koskevista kysymyksistä vaatimusmäärittelyvaiheen kysymykset voisivat olla kaikille yhteisiä ja suunnitteluvaiheen kysymykset ohjelmistokehittäjille suunnattuja.

"...sitten noille tuota kehittäjille olisi vähän semmoiset kehittäjän rooliin sopivammat kysymykset ja sitten olisi tavallaan tämmöiselle prosessihenkilöille, joita voi olla muunlaisia rooleja kuin minä, niin olisi sitten toisen tyyppisiä kysymyksiä."

"Että jos haluaa tehdä semmoisen kyselyn et ikään kuin voisi olla teknisille asiantuntijoille, koodareille kohdennettu, silloin siinä voisi olla esimerkiksi just näitä ohjelmiston elinkaarimallin kysymyksiä."

Osa haastateltavista oli sitä mieltä, että kyselylomaketta ei tulisi jakaa osa-alueisiin. Tätä perusteltiin erityisesti sillä, että kaikkien ohjelmistokehittäjien tulisi tietää vihreään ohjelmistokehitykseen vaikuttavista asioista kokonaisvaltaisesti. Tätä perusteltiin myös sillä, että erillisistä osioista ei katsottu saavutettavan hyötyä tai niitä ei pidetty tarpeellisina.

"Mutta sitten taas, kun katselee täältä vähän pidempää, niin mun mielestä ei tää ole semmoinen, että pystyy (jakamaan osa-alueisiin). Siellä taisi jossain kysymyksessäkin olla vähän semmoista, että ei tätä pysty vaan ottaa semmoista pientä palaa ja olla tyytyväinen elämäänsä, että nyt mä teen vihreätä koodia mun mielestä kyllä tää pitäisi ymmärtää laajemmin, että kyllä sen kehittäjän pitää ymmärtää testauksenkin asiat ja päinvastoin."

Yksi haastateltavista ei osannut sanoa pitäisikö kyselylomake jakaa osa-alueisiin.

"En tiedä (olisiko hyvä, että kyselylomakkeessa pääsee hyppäämään johonkin tiettyyn aihealueeseen)."

Haastateltavien mielestä kyselylomakkeen alkuun tulisi lisätä hakemisto kyselylomakkeen sisällöstä ja kyselylomakkeen alussa tulisi kertoa kyselylomakkeen tarkoituksesta.

"...tälleen kun reflektoin tuota vähän taaksepäin niin siitä olisi saanut semmoisen vähän table of contents, mikä tää nyt on, hakemisto/luettelo tyyppisen."

"Että se briiffi tähän kysymykseen vastaamiseen on tärkeätä tuoda esiin, että ettei kukaan jätä vastaamatta siksi, että se pelkää väärää miinuspiste vastausta."

5.3.5 Kyselyn kieliasu ja termistö

Haastateltavien mielestä kyselylomakkeessa käytetty kieliasu on ymmärrettävä, mutta kyselylomakkeessa käytettyjä termejä tulisi selittää tarkemmin ja kysymyksiä tulisi avata enemmän.

"Joo kyllä mä sanoisin, että se on (ymmärrettävä)."

"Muuten joo tosiaan, mutta sitten just noissa joissakin kysymyksissä on se, että haluaisi vähän ehkä jotenkin enemmän kontekstia tai jotain tarkennuksia siihen joko johonkin termiin tai sitten siihen, että mitä siinä nyt sitten ihan tarkalleen tarkoitetaan."

"Niin joo (kysymyksiä pitäisi avata enemmän)."

Tämän lisäksi kyselylomakkeessa tulisi suosia englanninkielisiä termejä suomennosten sijasta.

"Osa oli vieraita sen takia, että ne oli suomeksi ja sitten taas ehkä on enemmän tottunut englanninkielisiin termeihin tällä alalla."

Lisäksi haastateltavat listasivat useita yksittäisiä termejä, jotka olivat epäselviä ja joita tulisi selittää tarkemmin. Näitä termejä olivat esimerkiksi sähköinen jäte sekä kevyt asiakaspääte.

"Niin joo itseasiassa tää viimeinen tää sähköinen jäte niin tää oli kyllä vähän semmoinen, että mä en ehkä ihan sisäistänyt täysin."

"...niin siis tuossa mä just jäin miettii sitä, että mikä tässä nyt on siis kevyt asiakaspääte."

5.3.6 Sähköinen kysely

Suurimman osan haastateltavista mielestä kyselylomake tulisi siirtää sähköiseen muotoon.

"Juu kyllä näin, että se (sähköinen kyselylomake) olisi kyllä oikein hyväkin juttu."

"Juu, se (sähköinen kyselylomake) varmasti kyllä parantaisi tavallaan tätä."

Yksi haastateltavista oli sitä mieltä, että sähköisen kyselylomakkeen sijaan tuotetta pitäisi kehittää verkkokurssiksi, jossa informatiivinen osuus on ennen kysymyksiä.

"Mutta mä tiedä onko se lopputulema sitten enemmän joku tämmöinen e-learning -kurssi kuin ihan vaan kyselylomake."

Sähköisessä kyselylomakkeessa tulisi olla myös kyselyn edistymisestä kertovat graafinen indikaattori.

"Varmaan, jos siitä tulee web-pohjainen, niin siinähän voisi olla hyvä semmoinen progressbar väintään kertomaan sitä, että missä kohtaa ollaan menossa siinä kyselyssä, että käyttäjä vähän tietää, että kuinka paljon vielä tässä on jäljellä."

5.3.7 Vihreyden toteutuminen organisaatiossa

Haastateltavat kertoivat ristiriitaisia mielipiteitä siitä, toteutuuko vihreys heidän organisaatiossaan. Osa haastateltavista oli sitä mieltä, että vihreys ei näy heidän organisaatiossaan. Heidän kokemuksensa mukaan vihreydestä työssä ei ole aikaisemmin puhuttu ja eivätkä he esimerkiksi osanneet sanoa onko vihreys mainittu yritysstrategiassa tai onko yrityksellä vihreydestä vastaavaa henkilöä. Heidän kokemuksensa mukaan vihreys ei näy myöskään asiakasrajapinnassa. Yksi haastateltavista kertoi vihreyden toteutumisen esteeksi ajanpuutteen. Toisaalta haastateltavat olivat sitä mieltä, että vihreys näkyy, mutta se ei ole selkeää tai ei esiinny terminä vihreys.

"Mä sanon ehkä provosoivasti, että töissä se ei näy, koska edellä mainituista syistä en koe että olisi semmoista kulttuuria, että tai työtapoja tai työmenetelmiä, että ainakaan sen

kautta ajateltaisiin asioita."

"Niin ei se sillä lailla terminä kulje mukana meillä eikä asiakkaalla vielä ainakaan."

Toisaalta haastateltavat olivat sitä mieltä, että vihreys näkyy jollakin tavalla organisaatiossa ja he kertoivat useita esimerkkejä siitä, miten vihreys näkyy heidän työssään. Esimerkiksi työntekijät pyrkivät hyödyntämään etätyöskentelymahdollisuuksia. Ohjelmistot pyritään rakentamaan optimoidusti siten, että ne esimerkiksi käyttäisivät resursseja järkevästi. Ohjelmistokehittäjät pyrkivät tekemään uudelleenkäytettävää koodia ja arkkitehtuuriratkaisuista pyritään tekemään järkeviä. Vihreyden kerrottiin näkyvän asiakasrajapinnassa siten, että vaatimus vihreydestä tulee asiakkaalta ja oman organisaation vastuullisuusjohtaja toimii yhteistyössä asiakkaan kanssa. Vihreyden myös koettiin näkyvän organisaation infrastruktuurissa sekä esimerkiksi pakkausmateriaalien kierrättämisenä.

"Varmaan se konkreettisin omalta osalta miten se näkyy on matkustamisen minimointi. Ja kun keskustellaan siitä, että tarviiko joku palaveri järjestää livenä ja kun meilläkin porukkaa on, tässä palvelussa on neljällä paikkakunnalla on konttorit ja sitten vielä muutamia henkilöitä asuu pidemmällä vielä näistä konttoreista, niin pyritään minimoimaan sitä matkustamista..."

"...kun me tavallaan perinteisestikin on tehty jotain ohjelmaa, missä tehdään vaikka laskentaa, niin tietysti luontaisesti on yritetty optimoida tehdä se niin, että se olisi mahdollisimman nopea, että se (ohjelmisto) ei käyttäisi resursseja, joka jo sinällään on tavallaan sama asia kuin resurssiviisaus ja vihreä tässä kontekstissa vihreyteen pyrkiminen."

5.4 Ensimmäisen iteraation kehittämistuotos

Tässä osassa esitetään ensimmäisen iteraation kehittämistuotoksessa muuttuneet asiat. Kyselyalustaksi valikoitui yleisesti käytetty LimeSurvey-palvelu. Yllätyksenä kuitenkin tuli se, että LimeSurveyn toiminnallisuudet eivät olleet odotusten mukaiset ja monivalintakysymyksistä jouduttiin pisteidenlaskun vuoksi jättämään väärät vastaukset pois.

Kyselyyn lisätyt tai muuttuneet asiat esitetään lihavoidulla fontilla. Kyselystä poistetut asiat esitetään yliviivattuina. "Kyllä", "Ei"ja "En tiedä-vastausten perässä

suluissa esitetyistä numeroista numero yksi (1) tarkoittaa oikeaa ja numero nolla (0) väärää vastausta.

Vihreä ohjelmistokehitys ja vihreät työtavat

Tässä osiossa kysytään vihreisiin työtapoihin liittyviä kysymyksiä.

1. **Valitse oikeat vihreän ohjelmiston määritelmät.**

   (a) **Vihreän ohjelmiston suorat ja epäsuorat kielteiset vaikutukset talouteen, yhteiskuntaan, ihmisiin ja ympäristöön ovat mahdollisimman vähäiset. (1)**

   (b) **Vihreällä ohjelmistolla on myönteiset vaikutukset kestävään kehitykseen. (1)**

   (c) **Vihreä ohjelmisto on hiili- ja energiatehokas. (1)**

   (d) **Vihreä ohjelmisto käyttää sähköä pienellä hiili-intensiteetillä. (1)**

   (e) **Vihreät sovellukset käyttävät laitteistoa mahdollisimman tehokkaasti ja maksimoi laitteiston energiatehokkuuden. (1)**

   (f) **Vihreä ohjelmisto tuottaa mahdollisimman vähän dataa ja minimoi sen siirtelyn. (1)**

   Kaikki vastaukset ovat oikein. Vihreää ohjelmistoa ei pysty rakentamaan pelkästään tehokkailla algoritmeilla tai valitsemalla energiatehokasta ohjelmointikieltä. Vihreä ohjelmisto vaatii useiden eri näkökulmien huomioon ottamista aina vaatimusmäärittelystä ohjelmiston käytöstä poistamiseen asti.

2. Onko totta, että etätyöskentely vähentää hiilidioksidipäästöjä?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Etätyöskentely vähentää matkustamisesta syntyviä hiilidioksidipäästöjä.

3. Onko totta, että töihin pyöräily tai yleisten kulkuvälineiden käyttäminen vähentää hiilidioksidipäästöjä?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Pyöräily ja yleisten kulkuvälineiden käyttäminen vähentää hiilidioksidipäästöjä vähentämällä autoilua.

4. Onko totta, että paperitulostaminen ei lisää merkittävästi hiilidioksidipäästöjä?

   Kyllä (0) Ei (1) **En tiedä (0)**

   Paperitulostaminen lisää merkittävästi hiilidioksidipäästöjä.

5. Onko totta, että tietokoneen sammuttaminen tai lepotilaan laittaminen lyhyiksi ajoiksi, esimerkiksi kahvi- tai ruokatauon ajaksi ei vähennä merkittävästi tietokoneen energiankulutusta?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Yksittäisen tietokoneen sammuttaminen ei vähennä merkittävästi energiankulutusta. Sen sijaan, kun suuri joukko työntekijöitä sammuttaa tietokoneensa poistuessaan työpisteeltään esimerkiksi lounaalle, säästöt voivat kasvaa merkittävästi.

6. Onko totta, että etäyhteyksien avulla järjestettävät tapaamiset eivät vähennä hiilidioksidipäästöjä?

   Kyllä (0) Ei (1) **En tiedä (0)**

   Etäyhteyksien suosiminen vähentää matkustamisen määrää ja siten hiilidioksidipäästöjä.

7. Onko kevyiden asiakaspäätteiden **(eng. thin client)** käyttämisellä työtehtävissä merkitystä hiilidioksidipäästöjen vähentämisessä?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Kehitystyössä voidaan hyödyntää kevyitä asiakaspäätteitä, jotka käyttävät vain viidenneksen energiaa tavalliseen pöytätietokoneeseen verrattuna.

Vihreä ohjelmistokehitysprosessi

Tässä osiossa kysytään vihreään ohjelmistokehitysprosessiin liittyviä kysymyksiä.

1. ~~Onko tarkistuslistojen käytöstä hyötyä vihreässä ohjelmistokehitysprosessissa?~~

   ~~Kyllä~~ ~~Ei~~

2. Onko totta, että dokumentaation vähentäminen lisää ohjelmistokehitysprosessin ja ohjelmiston vihreyttä?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Turhan dokumentaation vähentäminen vähentää ajankäyttöä sekä muistin käyttöä ja siten se vähentää myös energiankäyttöä ja hiilidioksidipäästöjä.

3. ~~Onko totta, että jatkuvalla validoinnilla ei ole merkitystä ohjelmistokehitysprosessin vihreyden kannalta?~~

   ~~Kyllä~~ ~~Ei~~

4. Onko totta, että tehokas viestintä on eräs vihreän ohjelmistokehitysprosessin menestystekijöistä?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Tiimin ja asiakkaan viestintää voidaan edistää esimerkiksi valmentamalla tiimiä ennen projektin alkamista, kasvokkain tapahtuvilla tapahtumilla, verkon välityksellä tapahtuvalla viestinnällä, pitämällä tiimin koko mahdollisimman pienenä, epävirallisilla tapaamisilla sekä sosiaalisilla aktiviteeteilla.

5. ~~Jos ohjelmistokehitystiimissä on vihreästä ohjelmistokehityksestä vastaava henkilö, mitkä ovat hänen vastuulla olevat asiat? Voit valita useita.~~

   (a) ~~Vihreästä ohjelmistokehityksestä vastaava henkilö on vastuussa prosessin ja ohjelmiston vihreyden mittaamisesta.~~

   (b) ~~Vihreästä ohjelmistokehityksestä vastaava henkilö ei ole vastuussa prosessin ja ohjelmiston vihreyden mittaamisesta, vaan vastuu on tiiminjohtajalla / Scrum Masterilla.~~

   (c) ~~Vihreästä ohjelmistokehityksestä vastaava henkilö on vastuussa prosessin ja ohjelmiston vihreyden dokumentoimisesta.~~

   (d) ~~Vihreästä ohjelmistokehitykestä vastaava henkilö ei ole vastuussa prosessin ja ohjelmiston vihreyden dokumentoinnista, vaan vastuu on jaettu tiimin kaikkien jäsenten kesken.~~

   (e) ~~Vihreästä ohjelmistokehityksestä vastaava henkilö on vastuussa prosessin ja ohjelmiston vihreyden raportoinnista.~~

(f) ~~Vihreästä ohjelmistokehityksestä vastaava henkilö ei ole vastuussa prosessin ja ohjelmiston vihreyden raportoinnista, vaan vastuu on tiiminjohtajalla / Scrum Masterilla.~~

6. **Onko totta, että ohjelmistokehitysprosessin vihreyttä ei tarvitse arvoida koko prosessin ajan vaan riittää, että prosessin vihreyttä arvioidaan kehittämisvaiheessa sekä testauksen aikana?**

   Kyllä (0) **Ei (1)** **En tiedä (0)**

   Ohjelmistokehitysprosessin vihreyttä tulisi arvioida koko prosessin keston ajan.

7. Alla on kaksi väittämää. Valitse näistä se, joka tukee paremmin vihreää ohejelmistokehitysprosessia.

   (a) Asiakkaan mukana oleminen vihreässä ohjelmistokehitysprosessissa koko prosessin keston ajan on tarpeellista. (1)

   (b) Asiakas on mukana vihreässä ohjelmistokehitysprosessissa vain vaatimusmäärittelyn aikana sekä ottamassa vastaan loppuraporttia. (0)

   Vihreän ohjelmistokehityksen näkökulmasta olisi hyvä, että asiakas on mukana tiiviisti koko ohjelmistokehitysprosessin ajan. Vaatimusmäärittelyn ja suunnittelun lisäksi asiakas on mukana myös jatkuvassa validoinnissa / hyväksymistestauksessa. Asiakas voi olla hyvinkin kiinnostunut ohjelmiston ja ohjelmistokehitysprosessin vihreyttä koskevista tuloksista, joten on perusteltua että myös hän saa raportin vihreyteen liittyvistä mittauksista ja kehityskohteista.
   Validoinnissa varmistetaan, että ohjelmisto täyttää asiakkaan sille asettamat odotukset. Ketterissä menetelmissä validointi tapahtuu yleensä iteraatioiden päätteeksi. Jatkuvalla validoinnilla tarkoitetaankin sitä, että validointi suoritetaan säännöllisesti, jotta ohjelmistosta tulee tarkoituksenmukainen ja mahdollisiin epäkohtiin päästään puuttumaan mahdollisimman varhaisessa vaiheessa.

8. ~~Alla on sprinttiin / iteraatioon liittyviä väittämiä. Valitse näistä oikeat.~~

   (a) ~~Sprintin kestolla ei ole vaikutusta ohjelmistokehitysprosessin vihreyteen.~~

(b) ~~Vihreän ohjelmistokehitysprosessin yksi menestystekijöistä on sprinttien / iteraatioiden pitäminen lyhyinä.~~

(c) ~~Sprintin loppuvaiheessa tulisi pitää vihreää ohjelmistokehitysprosessia ja ohjelmistoa koskeva arviointitapaaminen.~~

(d) ~~Sprintin / iteraation loppuvaiheessa ei tarvitse pitää vihreää ohjelmistokehitysprosessia ja ohjelmistoa koskevaa arviointitapaamista, vaan vihreyttä koskeva arviointi tapahtuu yhdessä sprintin / iteraation arvioinnin yhteydessä.~~

(e) ~~Sprintin / iteraation arvioinnin yhteydessä tulisi käydä läpi myös ohjelmistokehitysprosessin ja ohjelmiston vihreyttä koskevat tulokset.~~

9. **Onko totta, että prosessin vihreyttä dokumentoiva päiväkirjaa voidaan käyttää tukemaan vihreää ohjelmistokehitysprosessia?**

   Kyllä (1) **Ei (0)** **En tiedä (0)**

   Prosessin vihreyttä dokumentoiva päiväkirja on lyhyt dokumentti, johon kirjataan prosessin vihreyttä koskevat arviot sekä parannusehdotukset. Tämän dokumentin päivittäminen on koko tiimin vastuulla.

10. ~~Onko totta, että ohjelmistokehitysprosessin päättyessä pidettävä ohjelmiston vihreyttä koskeva retrospektiivi on hyvä tapa lisätä tulevien projektien vihreyttä?~~

    ~~Kyllä~~ ~~Ei~~

Vihreän ohjelmistokehitysprosessin mittaaminen

1. Onko totta, että vihreän ohjelmistokehitysprosessin tärkein mittari on hiilijalanjäljen mittaaminen?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Hiilijalanjälki määrittää, kuinka paljon hiilidioksidia ohjelmistokehitys-, hallinta- tai ylläpitovaihe tuottaa. Tämä on tärkein vihreä mittari ja lopulta kaikki vihreät tekijät pitäisi laskea sen avulla. Hiilijalanjälki on kuitenkin ongelmallinen siinä suhteessa, että sitä voidaan manipuloida esimerkiksi valitsemalla uusiutuvia energianlähteitä sähkön tuottamiseen.

2. Onko totta, että ohjelmistokehitysprosessin aikaisen / tuottaman hukan määrää voi myös mitata?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Hukkaa voi myös mitata. Ohjelmistokehitysprosessin aikainen hukka voi olla esimerkiksi energiahukkaa, käyttämätöntä kehitysaikaa tai jotain fyysistä hukkaa.

Vihreän ohjelmiston elinkaarimalli

Tässä osiossa kysytään vihreän ohjelmiston elinkaareen liittyviä kysymyksiä.

Vaatimusmäärittelyvaihe

1. Onko totta, että ohjelmiston vihreyttä koskevat vaatimukset tulisi asettaa tarkoilla määritelmillä?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Vihreyttä koskevat vaatimukset tulisi asettaa tarkoilla määritelmillä toiveiden, epätarkkojen tai vaillinaisten vaatimusten sijasta.

2. **Onko totta, että energiankulutusta koskevat vaatimukset tulisi ilmaista energiankäyttöön liittyvillä ilmaisuilla?**

   Kyllä (1) Ei (0) **En tiedä (0)**

   Energiankäyttövaatimukset tulisi ilmaista suoraan energiankäyttöön liittyvillä ilmaisuilla. Energiankäyttöön liittyviä ilmauksia ovat esimerkiksi Energia (E), Energiankulutus (kWh) ~~jne.~~

3. Onko totta, että epätarkat ja vaillinaiset vaatimukset voivat johtaa esimerkiksi siihen, että joutokäynnin **(eng. idle)** aikainen energiankulutus huomioidaan paremmin kuin suorituksenaikainen energiankulutus?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Epätarkat ja puutteelliset vaatimusmäärittelyt voivat johtaa siihen, että kaikkia ohjelmiston toimintoja ei huomioida ohjelmistokehityksen aikana. Joutokäynnin aikainen energiankulutus tulee huomioida, mutta huomiota tulee kiiinnittää myös ajonaikaiseen energiankulutukseen.

4. ~~Onko totta, että ohjelmistolle asetetut vaatimukset voivat olla ristiriidassa keskenään ja johtaa esimerkiksi siihen, että taloudelliset vaatimukset asetetaan ohjelmiston vihreyttä koskevien vaatimusten edelle?~~

   ~~Kyllä~~ ~~Ei~~

5. Mikä on vihreä arvioija? Valitse oikea.

   (a) Ohjelmisto, joka arvioi kehitettävän ohjelmiston energiankulutusarvion. (1)

   (b) Ohjelmisto, joka arvioi ohjelmistokehitysprosessin energiankulutusarvion. (0)

   (c) Ohjelmisto, joka arvioi kehittävän ohjelmiston hiilijalanjäljen. (0)

   Vihreä arvioija on ohjelmisto, joka arvoi vaatimusmäärittelyiden perusteella kehitettävän ohjelmiston energiankulutusarvion. Näitä ovat esimerkiksi Green Tracker tai Green Oracle https://dl-acm-org.ezproxy.jyu.fi/doi/pdf/10.1145/2901739.2901763

Suunnitteluvaihe

1. Onko totta, että säännöllisten suunnittelukokousten pitäminen on yksi vihreän ohjelmiston menestystekijöistä?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Suunnittelukokouksissa voidaan puuttua edelliskerran jälkeen havaittuihin epäkohtiin, tehostaa prosessia tai lisätä suunnitelmaan toimintoja joilla pyritään parantamaan ohjelmiston laatua ja vihreyttä.

2. Onko totta, että vihreän ohjelmistosuunnitelun näkökulmasta ohjelmistoja suunniteltaessa ei tarvitse ottaa huomioon vanhempia laitteistoja.

   Kyllä (0) Ei (1) **En tiedä (0)**

   Yksinkertainen keino edistää sovelluksen vihreytää on varmistaa ohjelmistojen ydintoimintojen toimiminen myös vanhemmissa laitteistoissa.

3. ~~Alla on suunnitteluvaiheeseen liittyviä väittämiä. Valitse näistä oikeat.~~

   (a) ~~Ohjelmistot tulisi suunnitella siten, että ne eivät kuluta ainakaan enempää energiaa kuin edellinen versio.~~

(b) ~~Ohjelmistoja suunniteltaessa ei tarvitse huomioida edellisen version energiankulutusta.~~

(c) ~~Ohjelmistot tulisi suunnitella siten, että ne eivät kuluttaisi ainakaan enempää muistia kuin edellinen versio.~~

(d) ~~Ohjelmistoja suunniteltaessa ei tarvitse huomioda edellisen version muistinkäyttöä.~~

(e) ~~Ohjelmistot tulisi suunnitella siten, että ne eivät kuluttaisi ainakaan enempää kaistanleveyttä kuin edellinen versio.~~

(f) ~~Ohjelmistoja suunniteltaessa ei tarvitse huomioida edellisen version kaistanleveyden kulutusta.~~

4. **Valitse oikea suunnitteluvaihetta koskeva väite.**

   (a) **Ohjelmistot tulisi suunnitella siten, että ne eivät kuluta ainakaan enempää energiaa kuin edellinen versio. (1)**

   (b) **Ohjelmistoja suunniteltaessa ei tarvitse huomioida edellisen version energiankulutusta. (0)**

   Ohjelmistot tulisi suunnitella siten, että ne käyttävät energiaa mahdollisimman tehokkaasti. Ohjelmistoa suunniteltaessa tulisi ottaa huomioon edellisen ohjelmistoversion energiankulutus ja pyrkiä parantamaan paljon energiaa kuluttavien toimenpiteiden tehokkuutta tai ainakin pyrkiä pitämään uusi ohjelmistoversio mahdollisuuksien mukaan samalla tasolla edellisen version kanssa energiankulutuksen suhteen.

5. **Valitse oikea suunnitteluvaihetta koskeva väite.**

   (a) **Ohjelmistot tulisi suunnitella siten, että ne eivät kuluttaisi ainakaan enempää muistia kuin edellinen versio. (1)**

   (b) **Ohjelmistoja suunniteltaessa ei tarvitse huomioda edellisen version muistinkäyttöä. (0)**

   Muistin käyttäminen käyttää myös energiaa. Uutta ohjelmistoversiota suunniteltaessa tulisi ottaa huomioon ohjelmiston muistin käyttö ja pyrkiä vähentämään turhaa muistinkäyttöä tai ainakin pitämään muistin käyttö mahdollisuuksien mukaan samalla tasolla edellisen version kanssa. Muistin käyttöä suunniteltaessa tulisi huomioida myös datan siirtelystä aiheutuvat kus-

tannukset. Jos datan siirtelyä ei voi välttää, sen voi yrittää ajoittaa sellaisiin ajanjaksoihin jolloin voidaan esimerkiksi hyödyntää enemmän vihreää sähköä ja pienentää siten hiilidioksidipäästöjä.

6. **Valitse oikea suunnitteluvaihetta koskeva väite.**

   (a) **Ohjelmistot tulisi suunnitella siten, että ne eivät kuluttaisi ainakaan enempää kaistanleveyttä kuin edellinen versio. (1)**

   (b) **Ohjelmistoja suunniteltaessa ei tarvitse huomioida edellisen version kaistanleveyden kulutusta. (0)**

   Suurten tietomäärien siirtäminen kuluttaa myös energiaa. Ohjelmistoja suunniteltaessa tulisi ottaa huomioon tietoliikenneyhteyksien käyttäminen ja huomioida kuinka usein ja paljon tietoa siirretään.

7. IoT-laitteiden (eng. IoT-devices) suunnittelussa ja toteutuksessa on paljon hyviä resursseja säästäviä periaatteita, joita voidaan hyödyntää myös muussa ohjelmoinnissa. Valitse näistä oikeat.

   (a) Ohjelmistot suunnitellaan siten, että ne eivät kulutua virtaa jatkuvasti. (1)

   (b) Ohjelmistot suunnitellaan siten, että ne eivät käytä jatkuvaa tietoliikenneyhteyttä. (1)

   (c) Ohjelmistot pyritään suunnittelemaan siten, että ne välttävät ylimääräistä muistinkäyttöä. (1)

   (d) Ohjelmistot pyritään suunnittelmaan siten, että ne käyttävät mahdollisimman tehokkaita algoritmeja. (1)

   IoT-laitteet suunnitellaan siten, että ne kuluttavat virtaa vain jaksottain pyrkimyksenä säästää rajallisia energiaresursseja.

   IoT-laitteet suunnitellaan siten, että tietoliikenneyhteyksiä käytetään vain tarvittaessa ja lähettämiseen käytetään mahdollisimman vähän energiaa. IoT-laitteet saattavat kerätä suuren määrän dataa, mutta sitä esikäsitellään ja vain kaikista oleellisin tieto lähetetään eteenpäin.

   IoT-laitteet suunnitellaan siten, että rajallista muistikapasiteettia käytetään vain tarvittaessa ja turhan tiedon säilyttämistä vältetään.

   IoT-laitteet suunnitellaan siten, että algoritmit käyttävät mahdollisimman vähän energiaa. Näin säästetään IoT-laitteiden rajallisia energiaresursseja.

Kehittämisvaihe

1. ~~Valitse oikeat väitteet koskien pariohjelmointia.~~

   (a) ~~Pariohjelmointi voi vähentää yksinkertaisten tehtävien suorittamiseen käytettävää aikaa.~~

   (b) ~~Pariohjelmointi voi lisätä yksinkertaisten tehtävien suorittamiseen käytettävää aikaa.~~

   (c) ~~Pariohjelmointi helpottaa monimutkaisten tehtävien ratkaisemista.~~

   (d) ~~Pariohjelmointi vaikeuttaa monimutkaisten tehtävien ratkaisemista.~~

   (e) ~~Pariohjelmoinnilla voidaan ratkaista monimutkaisia tehtäviä laadukkaammin.~~

   (f) ~~Pariohjelmoinnilla monimutkaisten tehtävien ratkaisut ovat usein huonolaatuisia.~~

2. Voidaanko uudelleenkäytettävällä koodilla vähentää energiankulutusta ja hiilidioksidipäästöjä?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Uudelleenkäytettävän koodin kirjoittaminen vähentää kehitystyöhön käytettävän ajan määrää ja siten se vähentää myös energiankulutusta ja hiilidioksidipäästöjä.

3. ~~Valitse oikeat väitteet koskien komponenttipohjaista kehitysstrategiaa.~~

   (a) ~~Komponenttipohjainen kehitysstrategia vähentää ajankäyttöä ja kustannuksia prosessissa.~~

   (b) ~~Komponenttipohjainen kehitysstrategia lisää ajankäyttöä ja kustannuksia prosessissa.~~

   (c) ~~Komponenttipohjainen kehitysstrategia mahdollistaa ohjelmiston osien uudelleenkäytön.~~

   (d) ~~Komponenttipohjainen kehitysstrategia estää ohjelmiston osien uudelleenkäytön.~~

   (e) ~~Komponenttipohjaisen ohjelmiston refaktoroinnon avulla voidaan selkeyttää ja tehostaa ohjelmistoa.~~

(f) ~~Komponenttipohjaisen ohjelmiston tehostaminen refaktoroinnin avulla on mahdotonta.~~

4. **Valitse oikea komponenttipohjaista kehitysstrategiaa koskeva väite.**

   (a) **Komponenttipohjainen kehitysstrategia vähentää ajankäyttöä ja kustannuksia prosessissa. (1)**

   (b) **Komponenttipohjainen kehitysstrategia lisää ajankäyttöä ja kustannuksia prosessissa. (0)**

5. **Valitse oikea komponenttipohjaista kehitysstrategiaa koskeva väite.**

   (a) **Komponenttipohjainen kehitysstrategia mahdollistaa ohjelmiston osien uudelleenkäytön. (1)**

   (b) **Komponenttipohjainen kehitysstrategia estää ohjelmiston osien uudelleenkäytön. (0)**

6. **Valitse oikea komponenttipohjaista kehitysstrategiaa koskeva väite.**

   (a) **Komponenttipohjaisen ohjelmisto refaktoroinnon avulla voidaan selkeyttää ja tehostaa ohjelmistoa. (1)**

   (b) **Komponenttipohjainen ohjelmisto tehostaminen refaktoroinnin avulla on mahdotonta. (0)**

Testausvaihe

1. Onko totta, että testitapausten suorittaminen testausstrategian mukaisesti vähentää energiankulutusta ja parantaa testauksen laatua?

   Kyllä (1 Ei (0) **En tiedä (0)**

2. Onko totta, että jatkuvan integraatioympäristön (eng. continuous integration) testipakettiin ei voi sisällyttää energia- ja suorituskykymittauksia?

   Kyllä (0) Ei (1) **En tiedä (0)**

   Jatkuvan integraatioympäristön testipakettiin voidaan sisällyttää myös energia- ja suorituskykymittauksia. Nämä tulokset voidaan näin toimittaa yhdessä muiden testitulosten kanssa kehittäjille sekä asiakkaalle.

3. **Onko totta, että raakaan voimaan (eng. brute force) perustuva testaus korreloi suoraan energiankulutuksen kanssa.**

   Kyllä (1) Ei (0) **En tiedä (0)**

   Raakaan voimaan perustuva testaus korreloi energiankulutuksen kanssa. Mitä enemmän raakaan voimaan perustuvaa testausta, sitä enemmän testaus kuluttaa energiaa.

Ylläpitovaihe

1. Onko totta, että ylläpitovaiheen vihreyteen liittyvät asiat tulisi ottaa huomioon jo tuotteen suunnittelu- ja kehittämisvaiheessa?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Ylläpitovaiheeseen liittyvät asiat tulisi ottaa huomioon ja tuotetta suunniteltaessa ja kehittäessä. Näissä vaiheessa voidaan huomioida esimerkiksi järjestelmän tuki- ja konfigurointitehtävät.

2. Onko totta, että ylläpitovaihe ei ole merkittävässä osassa ohjelmiston vihreyttä arvioitaessa?

   Kyllä (0) (1) **En tiedä (0)**

   Tuotteen hiilijalanjälki määritetään ohjelmistokehityksen, hallinnan ja ylläpitovaiheen aikaisen hiilidioksidipäästöjen avulla. Kaikki ohjelmiston ylläpitotoimet katsotaan tuoreeksi kehitystyöksi ja johtavat koko ohjelmiston elinkaariprosessin eli SDLC-prosessin toistamiseen.

3. Onko totta, että ylläpitovaiheen vihreyttä voidaan kasvattaa tarjoamalla käyttäjille energiansäästöohjeita sekä tapoja säätää ohjelmiston toimintaa energiaystävällisemmäksi?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Ohjelmiston käyttämän energian määrää voidaan vähentää tarjoamalla käyttäjille energiansäästöohjeita. Tämän lisäksi ohjelmiston tulisi olla rakennettu sellaiseksi, että käyttäjät voivat säätää ohjelmiston toimintoja tarpeidensa mukaan siten, että se kuluttaa vähemmän energiaa. Esimerkiksi käyttäjä voi säästää energiaa estämällä mainokset.

4. **Onko totta, että vain osa refaktorointitekniikoista vähentää ohjelmiston energiankulutusta?**

   Kyllä (1) **Ei (0)** **En tiedä (0)**

   Vain osa refaktorointitekniikoista vähentää myös ohjelmiston energiankulutusta. Ohjelmiston refaktorointitekniikkaa valitessa tulisi ottaa huomioon muiden laatutekijöiden lisäksi myös energiankulutuksen väheneminen.

5. **Onko totta, että ohjelmiston käytöstä poistamisen aikana ei voida kiinnittää huomiota vihreisiin näkökulmiin?**

   Kyllä (0) **Ei (1)** **En tiedä (0)**

   Ohjelmiston käytöstä poistamisen aikana voidaan ottaa huomioon esimerkiksi tiedostojen kierrätys tai niiden poistaminen kokonaan.

Vihreä ohjelmisto

Ohjelmiston vihreyden mittaaminen

1. ~~Onko totta, että vihreä ohjelmisto säästää resursseja ja vähentää hukkaa?~~

   ~~Kyllä~~ ~~Ei~~

2. Alla on listattu asioita, joilla mitataan ohjelmiston tehokkuutta. Valitse näistä oikeat.

   (a) CPU-intensiteetti (1)

   (b) Muistin käyttö (1)

   (c) Perifeerinen intensiteetti (1)

   (d) Joutokäynti (1)

   (e) Heijastuskyky (1)

   CPU-intensiteetillä mitataan kuinka monta CPU-jaksoa ohjelmisto kuluttaa. Jokaisen syklin vaatima energia on mitattavissa ja mukautettavissa muihin mittareihin.

   Muistin käytöllä voidaan seurata esimerkiksi päämuistin kulutusta. Tämän lisäksi seurataan kuinka muistia käytetään.

Perifeerisellä intensiteetillä seurataan sitä, kuinka paljon oheislaitteita käytetään. Tämä voidaan arvioida esimerkiksi laskemalla, kuinka monta pyyntöä ohjelmisto tekee oheislaitteille ja mitä resursseja tarvitaan. Toinen yksinkertainen tapa on laskea oheislaitteen energiantarve.

Joutokäynti ei koskaan ole hyödyllistä ja sillä hukataan resursseja. Joutokäynnillä siis seurataan, kuinka paljon ohjelmisto on käyttämättömänä.

Heijastuskyvyllä tarkoitetaan sitä miten ohjelmiston käyttäytyminen heijastuu muihin toimintoihin. Esimerkiksi muistiin perustuva ohjelmisto käyttää resursseja yleensä silloin kun tietokone käynnistetään. Tämä hidastaa tietokoneen käynnistämistä ja siitä voi seurata se, että tietokoneen käyttäjä jättää tietokoneen herkemmin päälle.

3. **Onko totta, että kahta ohjelmistoa ei voida verrata kestävän kehityksen näkökulmasta, jos ne eivät ole samanlaisissa ohjelmistojärjestelmissä?**

   Kyllä (1) Ei (0) **En tiedä (0)**

   Kahta eri ohjelmistoa ei voida verrata kestävyyden suhteen elleivät ne ole samanlaisissa ohjelmistojärjestelmissä. Kestävyyteen vaikuttavat tekijät ovat riippuvaisia järjestelmästä sekä toimialueesta, jossa ohjelmistoa suoritetaan.

4. ~~Onko totta, että kestävä kehitys voidaan määritellä tarkoitukseen sopivuudella, reduktiolla ja kauneudella?~~

   ~~Kyllä~~ ~~Ei~~

5. **Onko totta, että järjestelmäkutsujen profiloinnilla voidaan arvioida ohjelmiston energiankulutusta?**

   Kyllä (1) **Ei (0)** **En tiedä (0)**

   Järjestelmäkutsujen profiilin on osoitettu korreloivan ohjelmiston energiankulutuksen kanssa. Tämän takia järjestelmäkutsujen profiloinnilla voidaan epäsuorasti arvoida esimerkiksi ohjelmistoon tehtävien muutosten yhteydessä aiheuttavatko muutokset energiankulutuksen kasvua. Järjestelmäkutsujen profilointiin on kehitetty yksinkertaisia työkaluja ohjelmistokehittäjien avuksi.

Säästävät ohjelmistostrategiat

1. Onko totta, että säästävillä ohjelmistostrategioilla pyritään minimoimaan CPU:n ja muistin käyttö?

   Kyllä (0) Ei (1) **En tiedä (0)**

   Säästävillä ohjelmistostrategioilla pyritään minimoimaan virrankulutus. Kehittäjien tulisi adaptoida ohjelmointitekniikoita, jotka ovat mahdollisimman tehokkaita tiedon varastoinnin, laskennan sekä verkkoresurssien käytön suhteen.

2. **Onko totta, että laskentatehokkuudella voidaan määritellä ohjelmiston energiatehokkutta?**

   Kyllä (1) Ei (0) En tiedä (0)

   Laskentatehokkuudella voidaan määritellä ohjelmiston energiatehokkuutta. Laskentatehokkuutta voidaan parantaa esimerkiksi tehokkailla algoritmeilla, hyödyntämällä monisäikeistystä sekä vektorointia.

3. **Onko totta, että laskennan viivästyttämisellä tai laskennan tarkkuuden alentamisella ei ole merkitystä ohjelmiston energiankulutuksen kannalta?**

   Kyllä (0) Ei (1) En tiedä (0)

   Ohjelmiston energiankulutusta voidaan alentaa esimerkiksi alentamalla laskennan tarkkuutta ja viivästyttämällä sitä, jos ohjelmiston vaatimukset sen sallivat.

4. **Onko totta, että I/O-kutsujen (Input/Output-kutsujen) optimoinnilla ei ole merkitystä ohjelmiston energiankulutuksen kannalta?**

   Kyllä (0) Ei (1) En tiedä (0)

   Laskentatehokkuuden lisäksi ohjelmistoa rakentaessa tulisi ottaa huomioon datatehokkuus. Tämä tarkoittaa sitä, että datan liikkuminen operoinnin aikana pidetään mahdollisimman vähäisenä. I/O-kutsujen optimointi on yksi keino optimoida ohjelmiston toimintaa.

5. **Onko totta, että datatehokkuttaa ei voida parantaa muistihierarkian ja datastruktuurien hyvällä suunnittelulla?**

   Kyllä (0) Ei (1) En tiedä (0)

Algoritmien lisäksi muistihierarkian ja datastruktuurien hyvä suunnittelu sekä välimuistin tehokas käyttäminen ovat tärkeässä roolissa datatehokkuutta optimoitaessa.

6. **Onko totta, että rinnakkaislaskennalla ei voida pienentää ohjelmiston energiankulutusta?**

   Kyllä (0) **Ei (1)** **En tiedä (0)**

   Rinnakkaislaskennalla ja säikeistämisellä voidaan pienentää ohjelmiston energiankulutusta. Energiankulutusta on onnistuttu vähentämään kohtalaisesti jopa huonosti rinnakkaislaskentaan soveltuvissa ohjelmistoissa. Energiankulutuksen väheneminen johtuu ensisijaisesti suorittamiseen kuluvan ajan vähenemisestä ja siten prosessorin lepotilassa olevan ajan lisääntymisestä.

Pilvipalvelut ja reunalaskenta

1. Onko totta, että pilvipalvelujen käyttäminen lisää ohjelmiston energiankäyttöä?

   Kyllä (0) Ei (1) **En tiedä (0)**

   Pilvipalvelujen hyödyntämisellä on monia energiaa säästäviä ratkaisuja ja energiankulutusta voidaan vähentää sillä tavoin huomattavasti. Datan lisäksi pilveen voidaan sijoittaa laskentaa ja muita ohjelmiston ominaisuuksia. Pilveistämisellä on voitu säästää jopa 64 % energiankulutuksesta.

2. ~~Alla on listattu väitteitä siitä, miksi pilvipalvelut ovat hyödyllisiä ajateltaessa vihreää ohjelmistokehitystä. Valitse näistä oikeat.~~

   (a) ~~Pilvipohjaisissa palveluissa resursseja käytetään pyynnöstä, joka mahdollistaa isossa mittaskaalassa datakeskusten energiatehokkuuden optimoinnin.~~

   (b) ~~Pilvipohjaisissa palveluissa resursseja käytetään pyynnöstä, joka mahdollistaa asiakaslaitteiden energiatehokkuuden optimoinnin.~~

   (c) ~~Pilvipalveluissa on vähemmän yleiskustannuksia sekä tehokkaampi skaalautuvuus.~~

   (d) ~~Pilvipalveluissa on enemmän yleiskustannuksia, mutta tehokkaampi skaalautuvuus.~~

(e) ~~Datakeskukset voivat tehokkaammin hyödyntää vihreitä energianlähteitä ja laitteiden lämpenemisestä syntyvän hukkaenergian.~~

(f) ~~Laitteiden joutokäyntiä on vähemmän.~~

3. **Valitse oikea pilvipalvelujen hyödyllisyyttä koskeva väite.**

   (a) **Pilvipohjaisissa palveluissa resursseja käytetään pyynnöstä, joka mahdollistaa isossa mittaskaalassa datakeskusten energiatehokkuuden optimoinnin. (1)**

   (b) **Pilvipohjaisissa palveluissa resursseja käytetään pyynnöstä, joka mahdollistaa asiakaslaitteiden energiatehokkuuden optimoinnin. (0)**

4. **Valitse oikea pilvipalvelujen hyödyllisyyttä koskeva väite.**

   (a) **Pilvipalveluissa on vähemmän yleiskustannuksia, tehokkaampi skaalautuvuus sekä joutokäyntiä on vähemmän.(1)**

   (b) **Pilvipalveluissa on enemmän yleiskustannuksia, mutta tehokkaampi skaalautuvuus sekä joutokäyntiä on vähemmän. (0)**

5. **Valitse oikea pilvipalvelujen hyödyllisyyttä koskeva väite.**

   (a) **Datakeskukset voivat tehokkaammin hyödyntää vihreitä energianlähteitä ja laitteiden lämpenemisestä syntyvän hukkaenergian. (1)**

   (b) **Datakeskukset hyödyntävät huonosti vihreitä energianlähteitä ja laitteista syntyvää hukkaenergiaa. (0)**

6. Pelkästään pilvipalvelujen käyttäminen ei takaa kuitenkaan vihreämpää ohjelmistoa. Mitä ohjelmistokehittäjän tulee ottaa huomioon, jotta pilvipalvelujen käyttämisestä saadaan oletettu hyöty? Valitse oikeat.

   (a) Ohjelmistoarkkitehtuuri tulee suunnitella pilvipalvelujen käyttöä ajatellen. (1)

   (b) Datan siirtely tulee optimoida. (1)

   (c) Kaksoisdata tulee pyrkiä poistamaan. (1)

   (d) Ohjelmistokehittäjän tulisi hyödyntää älykästä tietojen pakkaamista datan käyttötiheyteen perustuen. (1)

   (e) Ohjelmiston tulisi hyödyntää myös palvelitonta laskentaa. (1)

(f) Ohjelmiston tulisi hyödyntää Faas-toimintoa (function-as-service). (1)

(g) Ohjelmistossa tulisi tarpeen mukaan hyödyntää datan käytön profilointia. (1)

Ohjelmistoarkkitehtuuri voi vaatia suuriakin muutoksia.

Palveliton laskenta ja Faas-toiminto mahdollistavat resurssien saumattoman skaalautumisen.

Datan käytön profiloinnilla ja tekoälypohjaisilla profilointisovelluksilla voidaan lötyää käyttäjien toimintamalleja ja säätää sovelluksen toimintaa tarpen mukaisesti ja lisätä siten tehokkuutta.

Ohjelmiston datan käsittely ja siirtely tulisi hoitaa mahdollisimman tehokkaasti. Näillä toimilla pyritään vähentää varastoitavan datan määrää, laskentaresurssien tarvetta sekä tietoliikenneresurssien tarvetta.

Ohjelmointikieli

1. Onko totta, että ajallisesti nopein ohjelmointikieli on myös tehokkain?

   Kyllä (0) Ei (1) **En tiedä (0)**

   Kulutettu energia ei ole suoraan verrannollinen kulutettuun aikaan ja tulokset ovat riippuvaisia kielestä ja käsiteltävänä olevasta ongelmasta.

2. Onko totta, että tulkitut kielet ovat energiatehokkaampia muihin kieliin verrattuna?

   Kyllä (0) Ei (1) **En tiedä (0)**

   Kielen energiatehokkuus on riippuvainen siitä ohjelmointiongelmasta, mihin sitä aiotaan käyttää. Regex-redux vertailussa tulkitut kielet vaiktutava olevan energiatehokkaampi valinta, vaikka tulkitut kielet eivät ole kovinkaan energiatehokkaita muita skenaarioita tarkasteltaessa.

3. Onko totta, että on mahdollista valita yksi ohjelmointikieli, joka on parempi muihin ohjelmointikieliin verrattuna?

   Kyllä (1) Ei (0) **En tiedä (0)**

Energian ja ajankäytön optimoimisen suhteen parhaimman ohjelmointikielien valitseminen on mahdollista. Sen sijaan ajan- ja muistinkäytön, energian- ja muistinkäytön sekä energian-, ajan- ja muistinkäytön suhteen saadaan muodostettua kielijoukkoja, joiden jäsenet ovat saman arvoisia toistensa kanssa.

4. Onko totta, että C-pohjaiset kielet eivät ole kovinkaan energiatehokkaita?

   Kyllä (0) Ei (1) **En tiedä (0)**

   C-pohjaiset kielet ovat yleisesti ottaen energian, ajan ja muistin suhteen tehokkaimpia. Muita hyvin menestyneitä kieliä ovat esimerkiksi Ada ja Rust.

~~Datan ja muistin käyttö~~

1. ~~Onko totta, että kokonaismuistin määrän ja DRAM-muistin energiankulutuksen välillä on suora yhteys?~~

   ~~Kyllä~~ ~~Ei~~

Sähköinen jäte

1. Onko totta, että sähköisellä jätteellä **(eng. electric waste)** voidaan tarkoittaa elektroniikkajätettä tai datajätettä?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Sähköinen jäte voidaan ymmärtää joko elektroniikkajätteenä tai datajätteenä, joka kuluttaa tietokone- ja henkilöstöresursseja. Sähköinen jäte lisää energiankäyttöä sekä hiilijalanjälkeä ja on siksi kriittinen menestystekijä vihreässä ohjelmoinnissa. Sähköistä jätettä syntyy vaatimusmäärittelyvaiheessa esimerkiksi silloin, kun tuotetta ei välttämättä oteta käyttöön ollenkaan, kun virheellisten vaatimusten takia koodia joudutaan kirjoittamaan uudestaan vastaamaan asiakkaan vaatimuksia, koodi tai dokumentaatio on liian monimutkaista tai yksityiskohtaista sekä kun ohjelmisto aiheuttaa ylimääräistä kuormitusta.

2. Onko totta, että sähköinen jäte ei ole merkittävä tekijä ohjelmiston hiilijalanjälkeä arvioitaessa?

   Kyllä (0) Ei (1) **En tiedä (0)**

Sähköinen jäte lisää energiankäyttöä sekä hiilijalanjälkeä, joten sitä voidaan pitää kriittisenä menestystekijänä virheässä ohjelmoinnissa.

3. Onko totta, että sähköistä jätettä syntyy vaatimusmäärittelyvaiheessa esimerkiksi silloin, kun vaatimukset ovat kirjattu virheellisesti ja ne eivät vastaa asiakkaan vaatimuksia?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Virheelliset vaatimukset johtavat siihen, että koodia joudutaan kirjoittamaan uudestaan vastaamaan asiakkaan vaatimuksia. Se voi johtaa myös turhien ominaisuuksien toteuttamiseen ohjelmistossa.

4. Onko totta, että monimutkainen koodi ja dokumentaatio lisäävät sähköistä jätettä?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Monimutkainen koodi ja monimutkainen sekä liian yksityiskohtainen dokumentaatio lisäävät sähköistä jätettä.

5. Onko totta, että ohjelmoijan puutteelliset taidot lisäävät sähköistä jätettä?

   Kyllä (1) Ei (0) **En tiedä (0)**

   Ohjelmoijan puutteelliset taidot lisäävät sähköisen jätteen määrää. Parhaiten sähköisen jätteen määrää voidaan vähentää toimimalla sovittujen mallien mukaisesti aluste lähtien.

Loppuviesti

Pääsit loppuun! Hienoa!
Toivottavasti löysit kyselystä uusia ajatuksia ja idoita! Loppujen lopuksi vihreässä ohjelmoinnissa on paljon sellaista, jota toteutetaan jo sellaisenaan tämän hetkisessä ohjelmistokehityksessä. Vihreä ohjelmisto kuitenkin vaatii sen, että vihreät näkökulmat otetaan huomioon ohjelmiston elinkaaren jokaisessa vaiheessa ja viedään hyväksi havaittuja asioita vielä pidemmälle. Lisäksi ohjelmisto ja ohjelmistokehitysprosessi vaatii kriittistä tarkastelua ja halua muuttaa epäekologisia toimenpiteitä enemmän ekologisiksi.

6 Toinen iteraatio

Tässä luvussa esitetään toisen iteraation kulku. Toisen iteraation tavoitteena oli kerätä palautetta kyselyn toisesta versiosta ja hyödyntää tätä palautetta kyselyn kolmatta versiota kehittäessä.

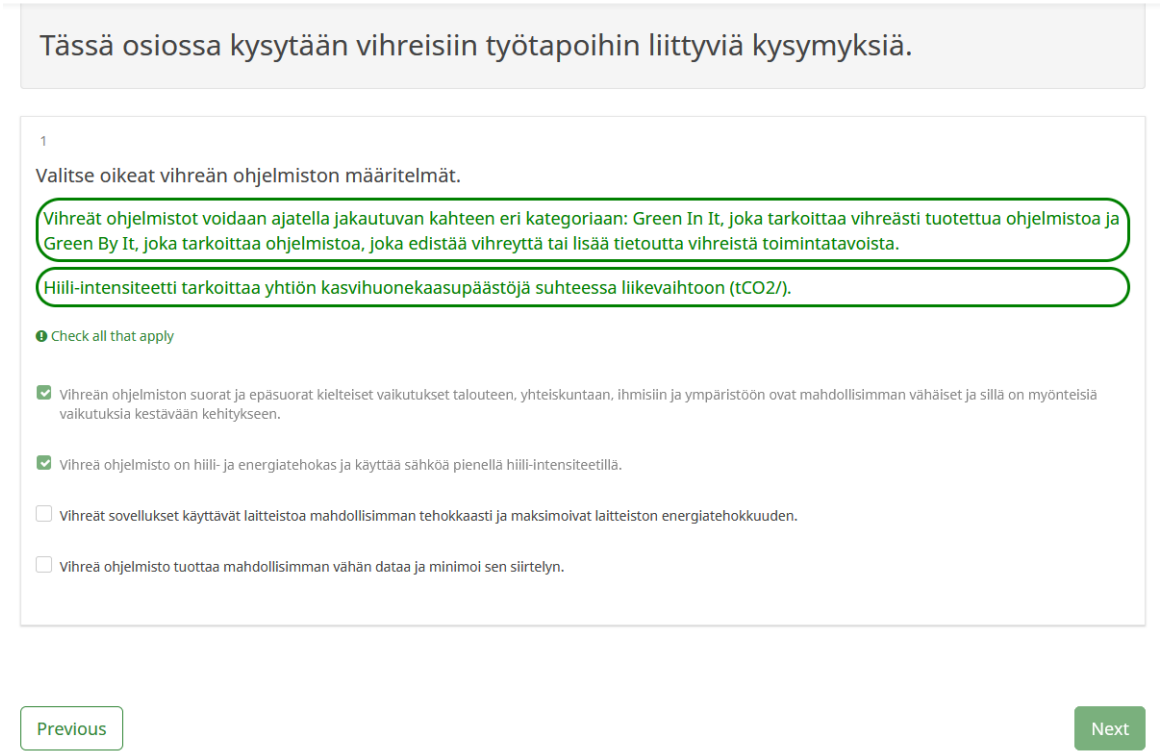
6.1 Toisen iteraation kulku

Tulosten perusteella kyselyn toisessa versiossa aihealueita tulisi muokata siten, että loppuyhteenvetoon lisätään linkkejä sivustoille, joista saa lisäinformaatiota vihreästä ohjelmoinnista. Tämän lisäksi loppuyhteenvetoa kehitetään siten, että se antaa kyselyn tekijälle palautetta siitä, millä kypsyystasolla hän on vihreän ohjelmointiosaamisen suhteen.

Kyselyn kysymykset käydään läpi siten, että yli neljä vastausvaihtoehtoa sisältävistä kysymyksistä poistetaan yli menevät vastausvaihtoehdot. Tästä esimerkki kuvassa 6.1. Samoin poistetaan sellaiset kysymykset, jotka ovat liian samankaltaisia jonkin toisen kysymyksen kanssa tai ovat muuten epäoleellisia eivätkä tuo siten lisäarvoa kyselyssä. Tällaisten kysymysten poistamisella pystytään myös lyhentämään kyselyä kuudestakymmenestä kysymyksestä viiteenkymmeneenviiteen kysymykseen.

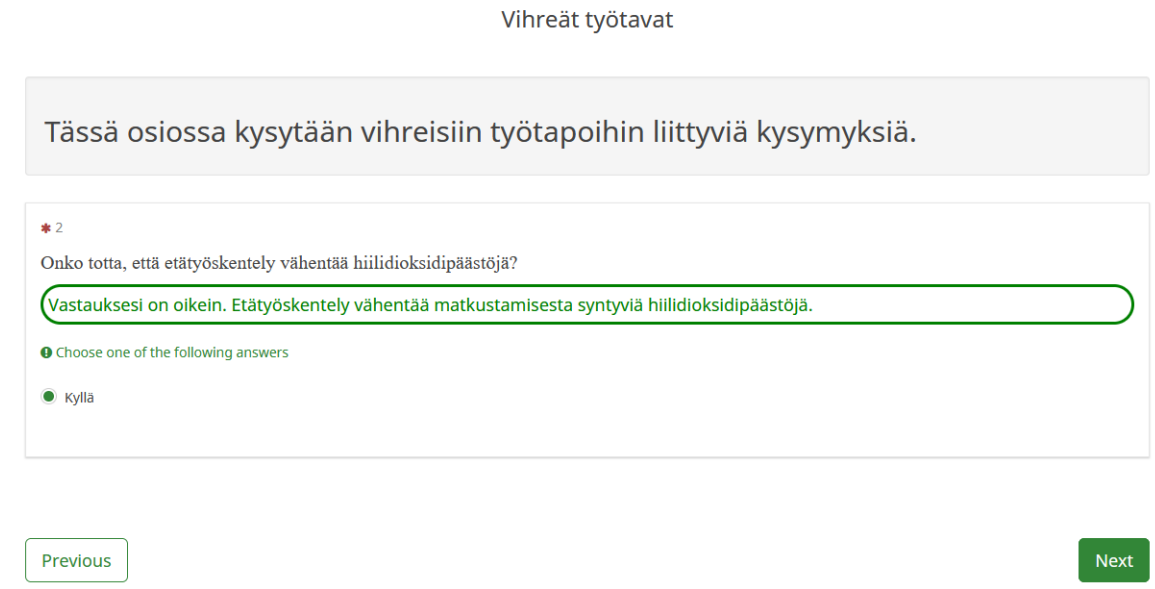
Kyselyn loppuun lisätään taustakysymyksiä ja esiintuotujen kysymysten kysymyksenasettelua tarkistetaan yksiselitteiseksi sekä kirjoitusvirheet korjataan.

Kyselyä ei jaeta roolipohjaisesti eri osioihin, koska työntekijöiden kokonaisvaltainen tuntemus ohjelmistoon vihreyteen vaikuttavista tekijöistä koettiin tärkeäksi asiaksi. Kyselyyn lisätään mahdollisuus palata takaisinpäin, jotta kyselyyn vastaava voi lukea informatiiviset osiot uudelleen niin halutessaan. Samoin vastaamisen aikaisia toimintoja muutetaan niin, että vastaaja ei voi siirtyä eteenpäin ennen kuin on checkbox-kysymyksissä vastannut kaikkiin tarvittaviin kohtiin.

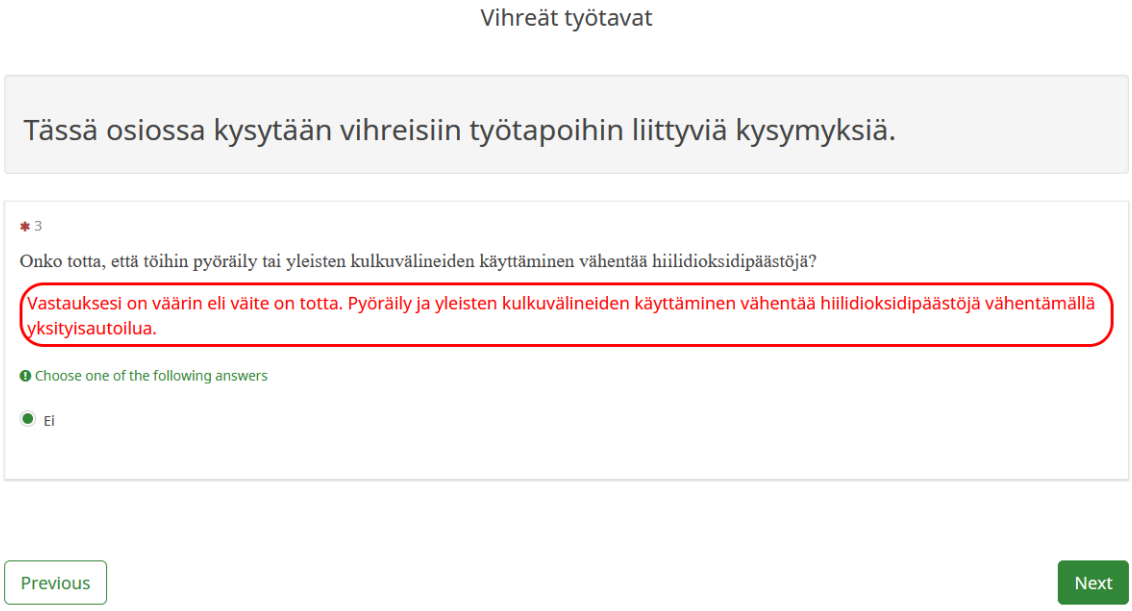


Kuva 6.1: Esimerkki monivalintakysymyksestä.

Informatiiviset osiot asetetaan kyselyssä korostetummin esille isontamalla fonttikokoa sekä asettamalla informatiivisen osuuden teksti kehyksen sisään. Aihealueiden vaihtumista korostetaan isontamalla aihealueen otsikon fonttikokoa. Informatiiviset osuudet värikoodataan siten, että oikeasta vastauksesta teksti esitetään vihreällä värillä vihreän kehyksen sisällä. Oikeasta vastauksesta ja infotekstistä esimerkki kuvassa 6.2. Väärästä vastauksesta teksti esitetään punaisella värillä punaisen kehyksen sisällä. Esimerkki väärästä vastauksesta ja infotekstistä kuvassa 6.3. "En tiedä-vastauksen teksti esitetään oranssilla värillä oranssin kehyksen sisällä. Tästä esimerkki kuvassa 6.4. Ensimmäisessä kysymyksessä on poikkeuksena sininen osio, jossa kerrotaan kaikkien vastausten olevan oikein.



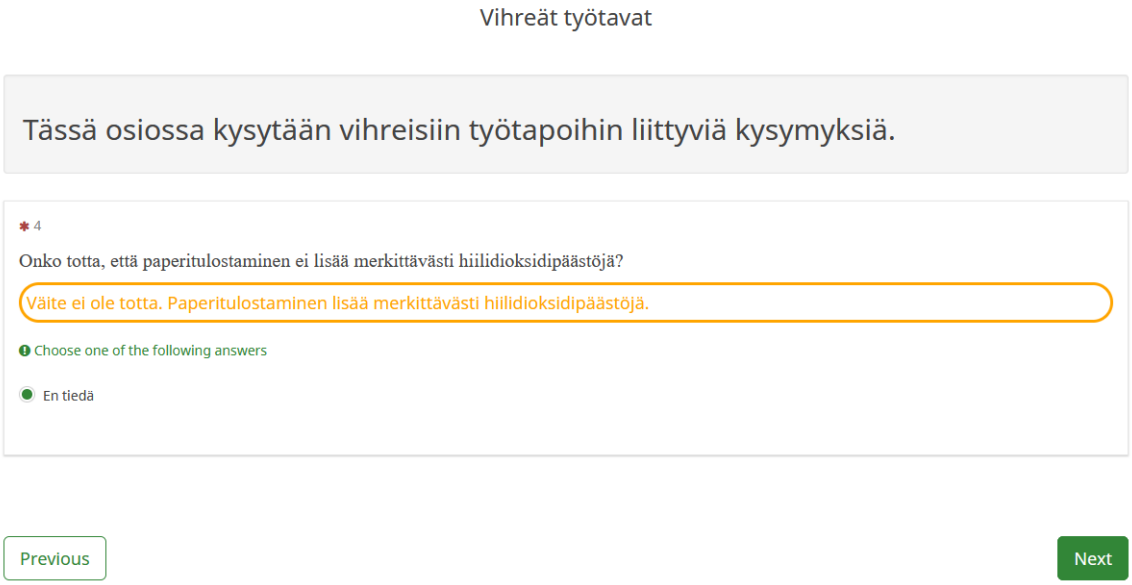
Kuva 6.2: Esimerkki "Oikein-vastauksesta.



Kuva 6.3: Esimerkki "Väärin-vastauksesta.

Kyselyn termit tarkistettiin ja termejä täsmennettiin tarvittavilta osin. Vastaamisen jälkeen esitettävistä informatiivisista osuuksista "En tiedä-vastauksen teks-

ti muutetaan neutraaliksi sekä kaikkiin kysymyksiin lisätään tarvittaessa puuttuva informatiivinen osuus.



Kuva 6.4: Esimerkki "En tiedä-vastauksesta.

Opetusmateriaaliin ja tarkistuslistaan liittyviä asioita ei kehitetä tässä tutkimuksessa eteenpäin, koska nämä aiheet vaatisivat oman tutkimustyön toteuttamisen. Kyselyyn ei myöskään lisätä faktoihin perustuvia laskelmia, koska näitä ei kirjallisessa osuudessa tullut esille.

6.2 Toisen haastattelun teemahaastattelun runko

Tässä osassa esitetään toisen teemahaastattelun runko. Haastattelun teemat toistavat ensimmäisen haastattelujen teemoja, mutta tarkentavia kysymyksiä on suppeammin.

1. Taustatiedot:
    (a) Onko haastateltava tutustunut etukäteen sähköiseen kyselyyn?
2. Kyselylomakkeen sisältö
    (a) Mitä mieltä haastateltava on kyselylomakkeesta?
    (b) Mitä mieltä haastateltava on kysymyksistä?

(c) Mitä mieltä haastateltava on kyselyn informatiivisista osuuksista?

(d) Mitä mieltä haastateltava on kyselylomakkeessa olevista aiheista?

3. Kysymysten formaatti

   (a) Mitä mieltä haastateltava on kysymyksenasettelusta?

   (b) Mitä mieltä haastateltava on vastausvaihtoehdoista?

4. Kyselylomakkeen ulkoasu

   (a) Mitä mieltä haastateltava on kyselylomakkeen ulkoasusta?

6.3 Toisen iteraation haastattelujen tulokset

Toisen haastattelun tuloksissa käsitellään haastateltavien yleistä mielipidettä kyselystä, kysymyksistä, kysymyksenasettelusta ja vastausvaihtoehdoista, ulkoasusta, pituudesta sekä informatiivisista osuuksista.

6.3.1 Haastateltavien yleinen mielipide kyselystä

Haastateltavat pitivät kyselyä yleisesti hyvänä ja kattavana. Heidän mielestään kysely tuntui tarkistuslistalta tai opetusmateriaalilta sekä formaattina hyvältä tällaiseen tarkoitukseen.

"On tosi kattava ja monipuolinen ja herättää varmasti miettimään kun noita väittämiä alkaa pohtimaan."

"...mutta kaikin puolin tykkäsin tästä, että tää oli tällainen, niin kun tämä opetti samalla kun siihen vastattiin niin siinä tuli semmoista, voiko sitä sanoa jopa vuorovaikutusta?"

Kyselyyn toivottiin luettavaksi ennakkomateriaalia ennen kyselyn läpikäymistä.

"Ja ehkä just muutenkin silleen vaikka nää tässä selittää sitten auki itseään, niin ehkä silti pitäisin tehokkaimpana sitä, että sitä olisi jonkinlainen pieni opetusmateriaali olemassa..."

Loppuyhteenvetoa toivottiin kehitettävän siten, että se kertoisi vastaajalle hänen oman tasonsa vihreän ohjelmoinnin suhteen sekä linkkejä lisäinformaation hankkimista varten.

"...voisiko siitä olla jotenkin johdettavissa jotain lisää tavallaan, että millä vaikka kypsyystasolla mä nyt oon tässä vihreässä ohjelmistokehitysprosessissa vaikka tai onko jotain mihin voisi kiinnittää huomiota jatkossa..."

Kyselyn hyödyntäminen sellaisenaan töitä tehdessä koettiin hankalana ja haastateltavat toivatkin esille, että kyselyn sisältämä tieto olisi helpompi hyödyntää tarkistuslistan muodossa.

"Mutta se, että koska tässä oli tosi paljon hyödyllisiä juttuja, niin sitten taas, jos joutuisi klikkailemaan tätä kyselyä läpi ja muistelemaan, että mitäs kaikkea tässä pitikään ottaa huomioon, niin sehän ei ole kätevää sitten taas sitten kun tekee oikeita töitä totta."

"Tuli ehkä semmoinen olo myös, että jos tän ensin tekee ja sitten alkaisi tällainen vihreän ohjelmistokehityksen projekti sen jälkeen, niin sitten olisi just kiva saada tästä jonkinlainen checklisti materiaali mitä sitten olisi helppo hyödyntää siellä tämän, kun on tämän ensin tehnyt tämän kyselyn, niin sitten voisi hyödyntää siinä oikeasti niitä töitä tehdä."

6.3.2 Kyselyn kysymykset

Kysymykset koettiin hyvinä, selkeinä, kuvaavina ja informatiivisina.

"Ne (kysymykset) on kyllä ihan hyviä."

"Ne (kysymykset) on tosiaan siis semmoisia, että sieltä uutta informaatiotakin tulee."

Osa kysymyksistä koettiin keskittymistä vaativana ja näihin vastaaminen vei enemmän aikaa kuin muihin kysymyksiin vastaaminen.

"Että niitä oli ehkä kaksi semmoista kysymystä, niin ne on ehkä se ainoa mikä jäi semmoiseen, että nää on vähän pitkiä, nää vaatii keskittymistä..."

Osassa kysymyksissä koettiin olevan liikaa vastausvaihtoehtoja ja samalla myös liikaa informaatiota kerralla.

"...mut ihan siellä loppusuoralla, on varmaan semmoinen missä voisiko olla 7-8:kin (vastausvaihtoehtoa) jopa?"

"Muutama kysymys niistä, mulla ei ole numeroa, jäi vähän semmoinen informaatioähkyä ne oli niitä kysymyksiä missä oli tosi paljon vastausvaihtoehtoja."

Kuva 6.5: Esimerkki monivalintakysymyksestä.

Haastateltavien mielestä joitain kysymyksiä voisi poistaa kyselystä. Tällaisten kysymysten koettiin olevan toistoa aikaisemmista kysymyksistä tai muuten epäolennaisia.

"...sanotaan ennen puolta väliä oli varmaan olisiko ollut 34. kysymys, että vähän täysin samaa aihepiiriä uudelleen ja uudelleen. Mietin sitä, että voiko niitä jotenkin yhdistää?"

Kyselyssä toivottiin olevan taustakysymyksiä.

"mutta ehkä itse asiassa tilastollisesti varmaan voisi olla hyvä kyllä saada se henkilön

rooli tohon, että missä roolissa toimii, niin sanottuna taustakysymyksenä, varmaan muunkinlaisia taustakysymyksiä tässä on ehkä hyvä olla."

6.3.3 Kysymysten kysymyksenasettelu ja vastausvaihtoehdot

Kysymysten kysymyksenasettelu koettiin hyvänä ja käänteisen kysymyksenasettelun koettiin pitävän vastaajan hereillä kyselyn aikana. Kysymystenasettelun koettiin olevan myös edelleen tenttimäisiä, mutta vastausten ja lisäinformaation saaminen vastaamisen jälkeen pehmensi kyselyä.

"...että se tavallaan kysymyksenasettelu käännetään välillä toisinpäin, niin mun mielestä se on kuitenkin ihan hyvä, että se pitää sen kyselyn täyttäjä vähän hereillä siinä sitten joo."

Vastausvaihtoehdot koettiin pääasiassa hyvänä. "En tiedä-vastausvaihtoehto jakoi mielipiteitä ja sitä pidettiin toisaalta tarpeellisena ja toisaalta sille ei nähty tarvetta.

"Kun itse tein omasta näkökulmasta, niin kyllä mä useampaan vastasin En tiedä ja mun mielestä se on ihan hyvä, että tavallaan ei aivan oikeasti nyt oikein tiedä tai näin, että se on ainakin hyvä, että siellä on sellainen vaihtoehto olemassa."

"No sekin ("En tiedä-vastaus) on tietysti validi vastaus varmaan sitten, että ehkä sitä en ymmärtänyt, että miksi se siellä on, mutta en siihen koskenut niin se ei sillä ei koskettanut mua."

Joitakin kysymyksiä tulisi muuttaa, koska näiden kysymysten kysymyksenasettelu antaa virheellisen kuvan käsiteltävänä olevasta asiasta.

"Mutta joo tosiaan siinä (kysymys nro. 14) ehkä se kysymyksenasettelu on semmoinen mitä voisi varmaan vielä viilata, koska se nyt antaa vähän liian semmoisen, että yksiselitteinen vähentäminen on hyvä ratkaisu."

6.3.4 Kyselyn ulkoasu

Kyselyn ulkoasua pidettiin hyvänä ja selkeänä.

"Se oli ihan hyvä aika silleen niin kun miellyttävää tosiaan, että ei niin kun oikeastaan mitään suurempaa kritiikkiä."

Kyselyn yläosassa sijaitseva kyselyn etenemisestä kertova indikaattori koettiin positiivisena asiana.
"Että mä tykkäsin siitä kun se oli se progressbar ylhäällä ja tiesin aika lailla just silleen että missä kohtaa ollaan menossa ja paljon jäljellä."

Kyselyn toiminnallisuuteen liittyviä kehittämiskohteita olivat mahdollisuus palata kyselyssä takaisinpäin sekä checkbox-kysymyksissä tulisi varmistaa se, että vastaaja ei siirry liian aikaisin seuraavaan kysymykseen.

"Joo yksi semmoinen mikä tuli ainakin mieleen oli toi, että siinä kyselyssä ei päässyt klikkaamaan takaisinpäin niihin vanhoihin vastauksiin, että siinä mielessä se on huono, että jos haluaisi lukea vaikka uudelleen jonkun tekstin sieltä niin sitten siihen ei pääse ja esimerkiksi just jos vahingossa klikkaan liian nopeasti eteenpäin niin silloin sitten tosiaan ei pääse tarkistamaan sitä."

"Joo sitten se oli ne monivalinta tai ne semmoiset checkbox valittavat, niin niissä oli tosiaan se huomio, että kun tsekkaa sen ensimmäisen checkboxin, niin sehän antaa sitten sen "vastaus on oikein"tyyppisen palautteen, niin siinä kyselyn käyttäjälle voi tulla se fiilis että "OK tää kysymys on nyt taputeltu"ja sitten se voi painaa sen jälkeen next ja sitten se lomake ei tavallaan anna siihen mitään palautetta, että siinä voi jäädä ihmisiltä mahdollisesti lukematta vaihtoehtoja."

Aihealueiden vaihtumista ja informatiivisia osuuksia haluttiin korostettavan.

"...että kun tulee niitä vastauksia sitten siihen tai niitä selityksiä näihin vastauksiin, niin ne saisi ehkä olla mun mielestä jotenkin selkeämmin jonkun vaikka freimin sisässä tai jollain vähän eri värikorotuksella, että ne huomaa silleen paremmin, että ne siihen ilmestyy."

Haastateltavat olivat pääasiassa sitä mieltä, että kyselyä ei ole tarpeellista jakaa

roolipohjaisesti ei osioihin. Yhden vastaajan mielestä kyselyn voisi jakaa roolipohjaisesti.

"Mä luulen että se vähän riippuu myös siitä, että jos jossain yrityksessä haluttaisiin kohdistaa tuo kysely tarkemmin jollekin kohderyhmälle, vaikka pelkästään koodareille, niin silloinkin sen pystyisi säätämään, että kyllä musta toi on hyvä, että se mieluummin noin kun spesifisti liian kapealle alueelle."

"...että voisiko olla niin kun eri rooleille vähän erityyppinen tai että pystyisikö tuosta roolipohjaisesti jotenkin karsimaan?"

6.3.5 Kyselyn pituus

Kyselyä pidettiin pääasiassa pitkänä. Toisaalta sitä pidettiin pitkänä, mutta ei liian pitkänä.

"Siinä mielessä se tietysti se pituus on tosiaan aika pitkä, että sitten jos niin kun tunninkin tuohon käyttää, niin onhan se silleen aika paljon..."

Haastateltavat olivat seuranneet kyselyn suorittamiseen kulunutta aikaa ja suoritusaika vaihteli kahdestakymmenestä minuutista ylöspäin. Yhden haastateltavan mielestä aiheen tunteminen etukäteen nopeutti kyselyyn vastaamista. Yksi haastateltavasta toi esille, että kyselyyn vastaajille tulisi antaa tarpeeksi aikaa kyselyyn vastaamiseen.

"Mulla meni semmoinen kaksikymmentä minuuttia, kun mä tein sen..."

"Ehkä tosiaan vaan se, että vastaajalle pitää kyllä antaa tarpeeksi aikaa."

6.3.6 Kyselyn informatiiviset osuudet

Haastateltavat kokivat kyselyssä olevat informatiiviset osuudet hyviksi ja riittäviksi. Oikean vastauksen ja perustelujen saaminen koettiin positiivisena asiana.

"Joo (informatiiviset osuudet ovat) siis tosi hyviä taustoittamaan just tavallaan niitä kysymyksiä..."

"Varmaan niitä ehkä, mutta sitten sekin on hyvä, että niihin vastauksiin tulee (informatiivisissa osuuksissa) tavallaan semmoinen perustelu."

Haastateltavat toivat esille, että joitain termejä tulisi täsmentää ja avata enemmän.

"No joissakin kohdissa mä silleen mietin, että niihinkin ehkä voisi vielä täsmentää jotakin (kysymyksissä)..."

"Mutta tosiaan jos siellä nyt jotakin termiä käytetään, niin tietysti silleen eihän se haittaa, jos sinne lisää auki kirjoitusta mahdollisesti joihinkin kohtiin."

Vastaamisen jälkeen esitettävä "en tiedä-vastausvaihtoehdon infoteksti tulee muuttaa neutraaliksi.

Kuva 6.6: Esimerkki ei-neutraalista "En tiedä-vastauksen palautteesta.

...silleen parempi olisi jos siihen en tiedä vaihtoehtoon tulisi sitten vaan tavallaan neutraalisti, että tässä on se oikea vastaus, eikä tosiaan sitä vastaus on väärin palautetta...

Haastateltavat toivat myös esille ensimmäisen kysymyksen informatiivisen tekstin puuttumisen sekä kyselystä löytyneitä kirjoitusvirheitä.

"...muutaman kirjoitusvirheen sieltä bongasin."

Yksi haastateltavista toi esille, että kyselyyn voisi lisätä faktoihin perustuvia laskelmia.

"...tän alan ihmiset tykkää tavallaan semmoisesta niin sanotusti faktoihin perustuvista tai että olisi jotain laskelmia."

6.4 Toisen iteraation kehittämistuotos

Tässä osassa esitetään toisen iteraation kehittämistuotoksessa muuttuneet asiat. Kyselyyn lisätyt tai muuttuneet asiat esitetään lihavoidulla fontilla ja kyselystä poistetut asiat esitetään yliviivattuina.

Vihreä ohjelmistokehitys ja vihreät työtavat

Tässä osiossa kysytään vihreisiin työtapoihin liittyviä kysymyksiä.

1. Valitse oikeat vihreän ohjelmiston määritelmät.
   (a) **Vihreän ohjelmiston suorat ja epäsuorat kielteiset vaikutukset talouteen, yhteiskuntaan, ihmisiin ja ympäristöön ovat mahdollisimman vähäiset, ja sillä on myönteisiä vaikutuksia kestävään kehitykseen. (1)**
   (b) ~~Vihreällä ohjelmistolla on myönteiset vaikutukset kestävään kehitykseen. (1)~~
   (c) **Vihreä ohjelmisto on hiili- ja energiatehokas ja käyttää sähköä pienellä hiili-intensiteetillä. (1)**
   (d) ~~Vihreä ohjelmisto käyttää sähköä pienellä hiili-intensiteetillä. (1)~~
   (e) Vihreät sovellukset käyttävät laitteistoa mahdollisimman tehokkaasti ja maksimoi laitteiston energiatehokkuuden. (1)
   (f) Vihreä ohjelmisto tuottaa mahdollisimman vähän dataa ja minimoi sen siirtelyn. (1)

Vihreät ohjelmistot voidaan ajatella jakautuvan kahteen eri kategoriaan: Green In It, joka tarkoittaa vihreästi tuotettua ohjelmistoa ja Green By It, joka tarkoittaa ohjelmistoa, joka edistää vihreyttä tai lisää tietoutta vihreistä toimintatavoista.

Hiili-intensiteetti tarkoittaa yhtiön kasvihuonekaasupäästöjä suhteessa liikevaihtoon (tCO2/).

Ohjelmiston tulisi pystyä käyttämään laitteistoa mahdollisimman tehokkaasti ja myös maksimoida laitteiston energiatehokkuuden.

Datan tuottaminen, säilöminen ja siirtäminen kuluttaa energiaa. Siksi vihreän ohjelmiston tulisi hyödyntää verkkoresursseja tehokkaasti.

Kaikki vastaukset ovat oikein. Vihreää ohjelmistoa ei pysty rakentamaan pelkästään tehokkailla algoritmeilla tai valitsemalla energiatehokasta ohjelmointikieltä. Vihreä ohjelmisto vaatii useiden eri näkökulmien huomioon ottamista aina vaatimusmäärittelystä ohjelmiston käytöstä poistamiseen asti.

2. Onko kevyiden asiakaspäätteiden (eng. thin client) käyttämisellä työtehtävissä merkitystä hiilidioksidipäästöjen vähentämisessä?

   Kyllä (1) Ei (0) En tiedä (0)

   Kehitystyössä voidaan hyödyntää kevyitä asiakaspäätteitä, jotka käyttävät vain viidenneksen energiaa tavalliseen pöytätietokoneeseen verrattuna.
   Kevyt asiakaspääte eli englanniksi "thin client"on riisuttu tietokone, jonka tarkoituksena on toimia vain yhteydenpitovälineenä isompaan ja tehokkaampaan keskustietokoneeseen.

Vihreän ohjelmistokehitysprosessin mittaaminen

1. **Onko totta, että ohjelmistokehitysprosessin tuottaman hukan määrää voidaan mitata?**

   Kyllä (1) Ei (0) En tiedä (0)

   Hukkaa voi myös mitata. Ohjelmistokehitysprosessin aikainen hukka voi olla esimerkiksi energiahukkaa, käyttämätöntä kehitysaikaa tai jotain fyysistä hukkaa.

Vihreän ohjelmiston elinkaarimalli

Tässä osiossa kysytään vihreän ohjelmiston elinkaareen liittyviä kysymyksiä.

Kehittämisvaihe

1. Valitse oikea komponenttipohjaista kehitysstrategiaa koskeva väite.

   (a) **Komponenttipohjainen kehitysstrategia vähentää ajankäyttöä ja kustannuksia prosessissa sekä mahdollistaa ohjelmiston osien uudelleenkäytön. (1)**

   (b) **Komponenttipohjainen kehitysstrategia lisää ajankäyttöä ja kustannuksia prosessissa, mutta mahdollistaa ohjelmiston osien uudelleenkäytön. (0)**

   Komponenttipohjainen kehitysstrategia vähentää ajankäyttöä ja kustannuksia sekä mahdollistaa koodin uudelleenkäytön. Uudelleenkäytettävien moduulien tulisi olla vain vähän riippuvaisia toisistaan, jotta ohjelmiston rakenne voidaan pitää mahdollisimman yksinkertaisena.

2. ~~Valitse oikea komponenttipohjaista kehitysstrategiaa koskeva väite.~~

   (a) ~~Komponenttipohjainen kehitysstrategia mahdollistaa ohjelmiston osien uudelleenkäytön. (1)~~

   (b) ~~Komponenttipohjainen kehitysstrategia estää ohjelmiston osien uudelleenkäytön. (0)~~

3. ~~Valitse oikea komponenttipohjaista kehitysstrategiaa koskeva väite.~~

   (a) ~~Komponenttipohjaisen ohjelmisto refaktoroinnin avulla voidaan selkeyttää ja tehostaa ohjelmistoa (1)~~

   (b) ~~Komponenttipohjaisen ohjelmiston tehostaminen refaktoroinnin avulla on mahdotonta. (0)~~

Testausvaihe

1. Onko totta, että testitapausten suorittaminen testausstrategian mukaisesti vähentää energiankulutusta ja parantaa testauksen laatua?

   Kyllä (1) Ei (0)

   Energiankulutuksen vähentämiseksi ja testauksen laadun parantamiseksi olisi tärkeää suorittaa testitapaukset testistrategian mukaisesti.

2. Onko totta, että raakaan voimaan (eng. brute force) perustuva testaus korreloi suoraan energiankulutuksen kanssa?

   Kyllä (1) Ei (0) En tiedä (0)

   Raakaan voimaan perustuva testaus korreloi energiankulutuksen kanssa. Mitä enemmän raakaan voimaan perustuvaa testausta, sitä enemmän testaus kuluttaa energiaa. Testauksessa olisi hyvä suosia niin sanottua älykästä testausta, jossa testausta kohdistetaan esimerkiksi aiemmin testaamattomiin koodisegmentteihin.

3. Alla on lueteltu väitteitä, joiden mukaan testausta voidaan tehostaa. Valitse näistä oikeat.

   (a) Staattisen koodianalyysin avulla voidaan tehostaa testausta. (1)

   (b) ~~Testitapaukset ja testien hallinta voidaan instrumentoida testien tehokkuuden tunnistamiseksi. (1)~~

   (c) Testattavuutta voidaan parantaa testivetoisella kehityksellä sekä testilähtöisen vaatimussuunnittelun avulla. (1)

   (d) ~~Testivetoisella kehityksellä sekä testilähtöisen vaatimussuunnittelun avulla voidaan helpottaa regressiotestausta ja tehdä siitä tehokkaampaa. (1)~~

   (e) Testauksen tehokkuutta voidaan parantaa kohdistamalla testausta aiemmin testaamattomiin koodisegmentteihin. (1)

   (f) Tekoäly voi auttaa määrittelemään mitkä testitapaukset on toteutettava ja missä määrin ja siten tehostaa testausta. (1)

   (g) ~~Älykäs validointi voi auttaa testitapausten valinnassa ja luomisessa ja siten tehostaa testausta. (1)~~

   Staattisen koodianalyysin avulla pyritään tunnistamaan ohjelmiston osia, jotka vaativat enemmän matalan tason testausta.

   Testilähtöinen ohjelmistokehitys ja testilähtöinen vaatimussuunnittelu helpottavat regressiotestausta ja tekevät siitä tehokkaampaa.

   Testien tehokkuutta voidaan arvioida instrumentoimalla testitapaukset ja testien hallinta.

   Tekoälyä voidaan hyödyntää myös testauksessa. Älykäs validointi voi auttaa testitapausten valinnassa ja luomisessa.

Ylläpitovaihe

1. **Valitse oikea refaktorointia koskeva väite.**

   (a) **Refaktoroinnin avulla voidaan selkeyttää ja tehostaa ohjelmistoa. (1)**

   (b) **Refaktoroinnin avulla voidaan selkeyttää ohjelmistoa, mutta ei tehostaa. (0)**

   Komponenttipohjaisen ohjelmiston refaktoroinnin avulla voidaan selkeyttää ja tehostaa ohjelmistoa. Refaktorointi tarkoittaa vihreässä asiayhteydessä lähdekoodin energiatehokkuuden optimointia muuttamatta lähdekoodin rakennetta.

   Refaktoroinnilla tarkoitetaan prosessia, jossa lähdekoodin sisäistä rakennetta muutetaan siten, että toiminnallisuus kuitenkin säilyy ennallaan. Muutokset voivat kohdistua esimerkiksi koodin luettavuuteen tai komponenttien työnjaon selkeyttämistä. Refaktoroinnin aikana ei lisätä ominaisuuksia tai pyritä lähtökohtaisesti korjaamaan ohjelmointivirheitä.

Vihreä ohjelmisto

Ohjelmiston vihreyden mittaaminen

1. Alla on listattu asioita, joilla mitataan ohjelmiston tehokkuutta. Valitse näistä oikeat.

   (a) CPU-intensiteetti (1)

   (b) Muistin käyttö (1)

   (c) Perifeerinen intensiteetti (1)

   (d) Joutokäynti (1)

   (e) ~~Heijastuskyky (1)~~

   CPU-intensiteetillä mitataan kuinka monta CPU-jaksoa ohjelmisto kuluttaa. Jokaisen syklin vaatima energia on mitattavissa ja mukautettavissa muihin mittareihin.

   Muistin käytöllä voidaan seurata esimerkiksi päämuistin kulutusta. Tämän lisäksi seurataan, kuinka muistia käytetään.

Perifeerisellä intensiteetillä seurataan sitä, kuinka paljon oheislaitteita käytetään. Tämä voidaan arvioida esimerkiksi laskemalla montako pyyntöä ohjelmisto tekee oheislaitteille ja mitä resursseja tarvitaan. Toinen yksinkertainen tapa on laskea oheislaitteen energiantarve.

Joutokäynti ei koskaan ole hyödyllistä ja sillä hukataan resursseja. Joutokäynnillä siis seurataan, kuinka paljon ohjelmisto on käyttämättömänä.

~~Heijastuskyvyllä tarkoitetaan sitä miten ohjelmiston käyttäytyminen heijastuu muihin toimintoihin. Esimerkiksi muistiin perustuva ohjelmisto käyttää resursseja yleensä silloin kun tietokone käynnistetään. Tämä hidastaa tietokoneen käynnistymistämistä ja siitä voi seurata se, että tietokoneen käyttäjä jättää tietokoneen herkemmin päälle.~~

2. ~~Onko totta, että kahta ohjelmistoa ei voida verrata kestävän kehityksen näkökulmasta, jos ne eivät ole samanlaisissa ohjelmistojärjestelmissä?~~

   ~~Kyllä (1)~~ ~~Ei (0)~~ ~~En tiedä (0)~~

   ~~Järjestelmäkutsujen profiilin on osoitettu korreloivan ohjelmiston energiankulutuksen kanssa. Tämän takia järjestelmäkutsujen profiloinnilla voidaan epäsuorasti arvioida esimerkiksi ohjelmistoon tehtävien muutosten yhteydessä aiheuttavatko muutokset energiankulutuksen kasvua. Järjestelmäkutsujen profilointiin on kehitetty yksinkertaisia työkaluja ohjelmistokehittäjien avuksi.~~

Pilvipalvelut ja reunalaskenta

1. Valitse oikea pilvipalvelujen hyödyllisyyttä koskeva väite.

   (a) Pilvipohjaisissa palveluissa resursseja käytetään pyynnöstä, joka mahdollistaa isossa mittaskaalassa datakeskusten energiatehokkuuden optimoinnin. (1)

   (b) Pilvipohjaisissa palveluissa resursseja käytetään pyynnöstä, joka mahdollistaa asiakaslaitteiden energiatehokkuuden optimoinnin. (0)

   Pilvipohjaisissa palveluissa resursseja käytetään pyynnöstä, joka mahdollistaa isossa mittaskaalassa datakeskusten energiatehokkuuden optimoinnin. Pilvipalvelut mahdollistavat esimerkiksi hyödyllisyyslaskennan, joka on esimerkki pyynnöstä tapahtuvasta laskennasta.

2. Valitse oikea pilvipalvelujen hyödyllisyyttä koskeva väite.

(a) Pilvipalveluissa on vähemmän yleiskustannuksia, tehokkaampi skaalautuvuus sekä joutokäyntiä on vähemmän. (1)

(b) Pilvipalveluissa on enemmän yleiskustannuksia, mutta tehokkaampi skaalautuvuus sekä joutokäyntiä on vähemmän. (0)

Pilvipalveluissa on vähemmän yleiskustannuksia, tehokkaampi skaalautuvuus sekä joutokäyntiä on vähemmän. Näiden asioiden lisäksi pilvipalvelimet voivat hyödyntää tehokkaammin vihreitä energianlähteitä sekä esimerkiksi kerätä laitteistoista vapautuvan hukkaenergian talteen.

3. ~~Valitse oikea pilvipalvelujen hyödyllisyyttä koskeva väite.~~

   (a) ~~Datakeskukset voivat tehokkaammin hyödyntää vihreitä energianlähteitä ja laitteiden lämpenemisestä syntyvän hukkaenergian. (1)~~

   (b) ~~Datakeskukset hyödyntävät huonosti vihreitä energianlähteitä ja laitteista syntyvää hukkaenergiaa. (0)~~

4. Pelkästään pilvipalvelujen käyttäminen ei takaa kuitenkaan vihreämpää ohjelmistoa. Mitä ohjelmistokehittäjän tulee ottaa huomioon, jotta pilvipalvelujen käyttämisestä saadaan oletettu hyöty? Valitse oikeat.

   (a) Ohjelmistoarkkitehtuuri tulee suunnitella pilvipalvelujen käyttöä ajatellen. (1)

   (b) **Datan siirtely tulee optimoida ja kaksoisdata poistaa. (1)**

   (c) ~~Kaksoisdata tulee pyrkiä poistamaan. (1)~~

   (d) Ohjelmistokehittäjän tulisi hyödyntää älykästä tietojen pakkaamista datan käyttötiheyteen perustuen. (1)

   (e) Ohjelmiston tulisi hyödyntää myös palvelitonta laskentaa. (1)

   (f) ~~Ohjelmiston tulisi hyödyntää Faas-toimintoa (function-as-service). (1)~~

   (g) ~~Ohjelmistossa tulisi tarpeen mukaan hyödyntää datan käytön profilointia. (1)~~

   Ohjelmistoarkkitehtuuri voi vaatia suuriakin muutoksia.

   ~~Palveliton laskenta ja Faas-toiminto mahdollistavat resurssien saumattoman skaalautumisen.~~

~~Datan käytön profiloinnilla ja tekoälypohjaisilla profilointisovelluksilla voidaan löytää käyttäjien toimintamalleja ja säätää sovelluksen toimintaa tarpeen mukaisesti ja lisätä siten tehokkuutta.~~

Ohjelmiston datan käsittely ja siirtely tulisi hoitaa mahdollisimman tehokkaasti. Näillä toimilla pyritään vähentämään varastoitavan datan määrää, laskentaresurssien tarvetta sekä tietoliikenneresurssien tarvetta.

Ohjelmistossa tulisi tarpeen mukaan hyödyntää datan käytön profilointia ja älykästä tietojen pakkaamista datan käyttötiheyteen perustuen.

Näiden lisäksi pitäisi hyödyntää palvelitonta laskentaa, joka mahdollistaa resurssien saumattoman skaalautumisen.

Sähköinen jäte

1. ~~Onko totta, että monimutkainen koodi ja dokumentaatio lisäävät sähköistä jätettä?~~

   ~~Kyllä (1)~~ ~~Ei (0)~~ ~~En tiedä (0)~~

   ~~Monimutkainen koodi ja monimutkainen sekä liian yksityiskohtainen dokumentaatio lisäävät sähköistä jätettä.~~

Taustakysymykset

1. **Koulutus**
   (a) **Lukio**
   (b) **Ammattikoulu**
   (c) **Ammattikorkeakoulu**
   (d) **Alempi korkeakoulututkinto**
   (e) **Ylempi korkeakoulututkinto**
   (f) **Ulkomailla suoritettu tutkinto**
   (g) **Muu**
2. **Sukupuoli**
   (a) **Nainen**
   (b) **Mies**

(c) **Muu**

(d) **En halua sanoa**

3. **Ikä**

(a) **Alle 25**

(b) **25–34**

(c) **35–44**

(d) **yli 45**

Loppuviesti

Pääsit loppuun! Hienoa!

Loppujen lopuksi vihreässä ohjelmoinnissa on paljon sellaista, jota toteutetaan jo sellaisenaan tämänhetkisessä ohjelmistokehityksessä. Vihreä ohjelmisto kuitenkin vaatii sen, että vihreät näkökulmat otetaan huomioon ohjelmiston elinkaaren jokaisessa vaiheessa ja viedään hyväksi havaittuja asioita vielä pidemmälle. Lisäksi ohjelmisto ja ohjelmistokehitysprosessi vaatii kriittistä tarkastelua ja halua muuttaa epäekologisia toimenpiteitä enemmän ekologisiksi.

Sait X pistettä vihreistä työtavoista!

9-10 pistettä = Mahtavaa! Osaat määritellä vihreän ohjelmiston ja tiedät tavallisimmat vihreät työtavat!

4-8 pistettä = Hienoa! Tiedät joitain vihreitä työtapoja ja tiedät suunnilleen, miten vihreä ohjelmisto määritellään. Toivottavasti löysit myös jotain uutta, jota voisit toteuttaa työssäsi!

0-3 pistettä = No höh! Tiedät vielä aika vähän vihreistä työtavoista ja ehkä vihreä ohjelmisto on sinulle uusi asia. Vihreät työtavat ovat melko yksinkertaisia tapoja vähentää energiankulutusta ja hiilidioksidipäästöjä työpaikalla, joten sinulle jäi varmasti mieleen joitain pääkohtia vihreistä työtavoista!

Sait X pistettä vihreää ohjelmistokehitysprosessia koskevista kysymyksistä!

6-7 pistettä = Mahtavaa! Tiedät paljon ohjelmistokehitysprosessin vihreyteen vaikuttavista asioista!

3-5 pistettä = Hienoa! Tiedät jo jotain ohjelmistokehitysprosessin vihreyteen vaikuttavista asioista! Toivottavasti löysit jotain uutta, jota voisi ehkä soveltaa seuraavassa projektissa!

0-2 pistettä = No höh! Ohjelmistokehitysprosessin vihreyteen vaikuttavat asiat ovat sinulle varmaan vielä uutta! Helppo tapa aloittaa ohjelmistokehitysprosessin vihreyden parantaminen on esimerkiksi edistää tiimin ja asiakkaan viestintää.

Sait X pistettä elinkaarimallia koskevista kysymyksistä!

20-27 pistettä = Vau! Tiedät jo todella paljon vihreän ohjelmiston elinkaarimalliin liittyvistä asioista! Jatka samaan malliin!

10-20 pistettä = Hienoa! Tiedät jo aika paljon vihreän ohjelmiston elinkaarimalliin liittyvistä asioista! Toivottavasti löysit jotain uutta, jota voisi ehkä toteuttaa seuraavassa projektissa!

0-10 pistettä = No höh! Ohjelmiston elinkaarimalliin liittyvät vihreät asiat ovat sinulle ehkä uutta. Elinkaarimallin näkökulmasta vaatimusmäärittely- ja suunnitteluvaiheella on iso vaikutus elinkaarimallin loppuvaiheiden vihreyteen. Voit kiinnittää huomiota esimerkiksi tulevan ohjelmiston virrankulutukseen, muistinkäyttöön, tietoliikenneyhteyksien käyttöön sekä pyrkiä suunnittelemaan tehokkaita algoritmeja. Näillä pääsee jo hyvään alkuun!

Sait X pistettä vihreää ohjelmistoa koskevista kysymyksistä!

19-26 pistettä = Mahtavaa! Tiedät jo todella paljon vihreään ohjelmistoon vaikuttavista asioista! Jatka samaan malliin!

8-18 pistettä = Hienoa! Tiedät jo aika paljon vihreään ohjelmistoon vaikuttavista asioista! Toivottavasti löysit jotain uutta, jolla voit parantaa seuraavan ohjelmiston vihreyttä!

0-7 pistettä = No höh! Vihreä ohjelmisto taitaa olla sinulle vielä aika uusi asia! Ohjelmistosta voi saada vihreämmän melko yksinkertaisillakin asioilla, kuten esimerkiksi valitsemalla ohjelmiston vaatimusten puitteissa tehokkaimman ohjelmointikielen ja pilveistämällä toimintoja, mutta osa toimenpiteistä voi olla vähän työläämpiä, kuten rinnakkaislaskenta. Helppo tapa aloittaa on esimerkiksi pyrkiä vähentämään ohjelmistoon liittyvää sähköistä jätettä.

Sait kyselystä yhteensä X pistettä!

55-70 pistettä = Vau! Tiedät jo todella paljon ohjelmiston vihreyteen vaikuttavista asioista! Olet tainnut ennenkin olla tekemässä vihreitä ja energiatehokkaita ohjelmistoja!

25-54 pistettä = Mahtavaa! Tiedät jo melko paljon ohjelmiston vihreydestä ja siihen vaikuttavista toimenpiteistä. Toivottavasti sait uusia ideoita, joita voisi toteuttaa seuraavassa projektissa!

0-24 pistettä = No höh! Vihreä ohjelmisto taitaa olla sinulle vielä melko uusi asia. Ohjelmiston vihreyteen voidaan vaikuttaa monilla erilaisilla toimenpiteillä ja valinnoilla. Kaikki toimenpiteet eivät sovi kaikkiin projekteihin, mutta useimpia voidaan soveltaa kuitenkin.

7 Tulokset

Tutkimuksen tavoitteena oli kehittää informatiivinen kysely, jolla voidaan myös mitata ohjelmistokehittäjien tietämystä vihreän ohjelmistokehitysprosessin ja vihreän ohjelmiston tekijöiden suhteen. Lopputuloksena syntynyt sähköinen kysely vastaa tätä tavoitetta. Kehittämisprosessin aikana pystyttiin luomaan informatiivinen kysely, joka koettiin opettavaisena sekä pääasiassa hyödyllisenä. Haastateltavat kokivat, että kysely on hyvä lähestymistapa parantamaan tietoisuutta vihreän ohjelmistokehityksen käytänteistä. Tietämyksen lisäämisen voidaan ajatella olevan hyödyllistä siinäkin mielessä, että jotkin ennestään käytössä olleet vakiintuneet käytänteet tehdä parempia ohjelmistoja tulevat perustelluksi ja paremmin ymmärretyiksi vihreyden kontekstissa. Kysely antaa kokonaisvaltaisen kuvan vihreään ohjelmistoon ja ohjelmistokehitysprosessiin vaikuttavista tekijöistä ja antaa käytännönläheisiä keinoja siihen, miten ohjelmiston ja ohjelmistokehitysprosessin eri osa-alueita voitaisiin tehdä vihreämmin. Kyselyn lopussa vastaajalle esitetään yhteenveto kyselyn tuloksista.

Digia Oyj:n Green Code Experttinä toimiva yhteyshenkilön arvio kyselystä oli positiivinen. Yhteyshenkilö piti kyselyä hyödyllisenä. Informatiivinen kysely otetaan käyttöön Digia Oyj:n sisäisessä koulutuksessa Digian omassa Moodle-alustassa. Kyselyä jatkokehitetään Digia Oyj:n omilla resursseilla ja tutkielmaa tullaan mahdollisesti hyödyntämään kokonaisuudessaan koulutusmateriaalin kehittämisessä.

Kirjallisuuskatsauksen perusteella pystyttiin selvittämään keskeisimmät vihreän ohjelmistokehityksen käytänteet. Ohjelmiston tuottamaan hiilijalanjälkeen voidaan vaikuttaa ohjelmistokehitysprosessin kautta, jolloin esimerkiksi vihreyttä koskevien huomioiden dokumentoinnilla, lyhyillä iteraatioilla ja asiakkaan tiiviillä mukana ololla voidaan vaikuttaa turhaan käytetyn työajan ja sähköisen jätteen määrään. Ohjelmiston elinkaarimallin kannalta vaatimusmäärittely- sekä suunnitteluvaihe korostuvat ohjelmiston vihreyden kannalta. Tässä vaiheessa pitää miettiä mitä ja miten lähdetään kehittämään, jotta ohjelmiston toiminta olisi mahdollisimman optimoitu ja tehokas. Vaatimusmäärittely ja suunnitteluvaiheessa valitut asiat ovat usein sellaisia, joita on vaikea perustavanlaatuisesti muuttaa enää myöhemmissä vaiheissa. Vaatimusmäärittelyvaiheessa tulisi kiinnittää huomiota erityisesti siihen, että asia-

kas ja yritys kommunikoivat tehokkaasti, jotta kehitetään varmasti asiakkaan haluamia asioita ja toisaalta, että ei kehitetä turhia asioita. Suunnitteluvaiheessa esimerkiksi arkkitehtuurisuunnittelulla ja energiatehokkaiden kielten valinnalla voidaan vaikuttaa valmistuvan ohjelmiston energiankulutukseen. Huomioitavaa on myös, että pilveistämisellä ja virtualisoinnilla on pystytty saavuttamaan huomattaviakin energiansäästöjä. Tässä vaiheessa pitäisi myös muistaa ottaa huomioon edellisissä projekteissa hyväksi havaitut asiat ja toisaalta välttää huonoiksi todettuja ratkaisuja.

Näkemys vihreiden käytänteiden toteutumisesta organisaatiossa vaihteli. Vastaus riippui myös siitä, kuinka pitkä ja laaja työkokemus haastateltavalla oli. Kokeneemmat ohjelmistokehittäjät osasivat nimetä vihreään ohjelmistokehitykseen liittyviä käytänteitä, vaikka vihreys terminä oli kaikille haastateltavalle melko uusi. Näitä olivat esimerkiksi optimointi, vältetään turhaa datan siirtelyä, yritetään tehdä uudelleenkäytettävää koodia ja pyritään muutenkin hyödyntämään resursseja järkevästi. Toisin sanoen vihreät käytänteet toteutuvat osittain organisaatiossa jo ennestään, vaikka haastateltavat eivät pitäneet näitä välttämättä vihreinä käytänteinä. Ja vaikka asioita kyllä tehdään vihreästi jo nyt, kaikki työntekijät eivät tiedä esimerkiksi ohjelmiston energiankulutukseen vaikuttavista asioista tai toimenpiteistä, joilla sitä voitaisiin vähentää. Laaja-alaisin näkemys asiasta oli sellaisilla haastateltavilla, joilla on pitkä työkokemus ohjelmistokehityksestä ja kokemusta useasta ohjelmiston elinkaarimallin vaiheesta.

Toisaalta ilmi tuli myös sellaisia asioita, joita organisaatiossa ei ollut vielä käytössä. Tällaisia asioita olivat esimerkiksi tarkistuslistojen käyttäminen tai ohjelmiston energiankäytön mittaamisen automatisoiminen jatkuvan integraatioympäristön avulla. Näin ollen tietämys vihreistä ohjelmistokehitysprosessiin ja ohjelmistokehitykseen liittyvistä käytänteistä lisääntyi ja tutkimuksen tulokset olivat näiltä osin myönteisiä.

8 Pohdinta

Tutkimuksen tulokset ovat linjassa aikaisempien tutkimusten kanssa. Haastateltavat kertoivat, että haastatteluhetkellä heillä ei ollut vielä käytössä vihreisiin käytänteisiin liittyviä ohjeistuksia. Osa haastateltavista ei esimerkiksi tiennyt, onko toimeksiantajan organisaatiolla vihreyteen tai kestävään kehitykseen liittyvää strategiaa tai onko heillä vihreydestä vastaavaa henkilöä. He kuitenkin suhtautuivat pääasiassa positiivisesti vihreyteen ja ajatukseen toteuttaa vihreämpiä sovelluksia.

Tutkimuksen tulokset ovat myös linjassa Ahmad et al. [6] tutkimuksen kanssa siinä, että vihreään ohjelmistokehitykseen liittyvät käytänteet vaihtelivat. Vihreiden käytänteiden hyödyntäminen oli joiltain osin puutteellista.

Suhtautuminen vihreyteen oli hyvin käytännönläheistä, myönteistä ja kustannusten hallinnan näkökulmasta ohjailtua. Tämä näkyi esimerkiksi pyrkimyksenä säästää energiaa. Ohjelmistotuotannossa ei myöskään vielä haastatteluhetkellä hyödynnetty esimerkiksi testaamisen aikaisia mittauksia. Olisikin mielenkiintoista tietää, voitaisiinko työntekijöitä motivoida vihreyden edistämiseen esimerkiksi palkitsemisella samalla tavalla kuin palkitaan esimerkiksi kouluttautumisesta tai muista ansioista? Tai pystyttäisiinkö ohjelmiston energiankulutuksen näkyväksi tuomisella paremmin motivoimaan työntekijöitä adaptoimaan energiaa säästäviä tekniikoita omaan työhön?

Tutkimus toteutettiin kehittämistutkimuksena ja kyselyn kehittämiseen tarvittava aineisto kerättiin teemahaastattelujen avulla, joten on aiheellista tarkastella tutkimusasetelmaa sekä kehittämistutkimuksen prosessin sekä teemahaastattelun ja sisällönanalyysin kannalta.

Tutkimuksen prosessia voidaan arvioida Hevner et al. [21] hyvän suunnittelutieteellisen tutkimuksen tutkimusohjeisiin. Tämän tutkimuksen lopputuloksena pystyttiin tuottamaan käytettävissä oleva kysely. Kysely luotiin ratkaisemaan Digia Oyj:n ongelma jakaa tietoa vihreistä käytänteistä ohjelmistokehityksessä omassa organisaatiossaan. Kehitetty kyselyä kehitettiin ja arvioitiin Digia Oyj:n henkilökuntaan kuuluvien haastateltavien toimesta sekä yhteyshenkilönä toimineen henkilön toimesta. Tutkimus on pyritty suorittamaan tarkasti ja tämä on pyritty osoittamaan raportoimalla prosessin kulku yksityiskohtaisesti ja selkeästi. Empiirisen osan suo-

rittamiseen on käytetty hyviä tieteellisiä käytäntöjä laadullisen tutkimuksen muodossa. Tutkimuksesta kirjoitetaan tämä tutkimusraportti, jossa käydään läpi prosessin kulku ja tutkimustulokset. Tarkalla raportoinnilla on pyritty parantamaan tutkimuksen toistettavuutta.

Näiden tutkimusohjeiden lisäksi tutkimusta voidaan arvioida prosessivaliditeetin näkökulmasta. Tutkimus toteutettiin vaiheittain ja johdonmukaisesti siten, että jokainen vaihe suoritettiin loppuun ennen seuraavan vaiheeseen siirtymistä. Tutkimusprosessissa toteutettiin kaksi iteraatiota ja edellisen vaiheen tuloksia hyödynnettiin seuraavassa iteraatiossa. Prosessin vaiheet on pyritty raportoimaan siten, että kehittämisprosessin iteraatiot ovat selkeästi esillä. Kehittämisprosessiin osallistui tutkimuksen tekijän lisäksi Digia Oyj:n Green Code Expert asiantuntijan roolissa. Hän myös oli varmistamassa, että tuotetta ollaan kehittämässä organisaation haluamaan suuntaan. Tuotteen arviointiin osallistui myös monipuolisesti kokemusta omaavia henkilöitä aina ohjelmistokehittäjistä ohjelmistoarkkitehteihin asti.

Luotettavuutta arvioidessa tulisi ottaa huomioon myös se, että validiteettia pitää arvioida sen kontekstin näkökulmasta missä artefaktia kehitetään ja siksi validiteetti määräytyy tapauskohtaisesti. Artefakti todettiin tutkimuksen aikana toimivaksi siinä kontekstissa, johon artefaktia oltiin kehittämässä. Tämä parantaa tulosten luotettavuutta.

Tutkimuksen luotettavuutta olisi voitu parantaa useammalla iteraatiolla. Tuotteen laatuun vaikuttaa myös se, että valittu ohjelmisto (LimeSurvey) ei toiminnallisuuksiltaan vastannut odotuksia ja monivalintakysymyksiin ei voitu sisällyttää vääriä vastauksia.

Useat kirjallisuuskatsauksessa hyödynnetyt tutkimukset ja niissä olevat toimintaperiaatteet vihreän ohjelmiston toteuttamiseksi ei ole välttämättä validoituja. Luotettavamman, validoituun tietoon perustuvan kyselyn luominen vaatisi paljon lisää tutkimuksia, jossa validoidaan kyseinen käytänteet. [37] Ongelmaksi muodostui myös se, että käytännönläheisiä ohjelmiston vihreyttä lisäävistä perusasioista kertovia uusia lähteitä ei ollut helposti löydettävissä ja kirjallisuuskatsauksessa on tämän takia käytetty suhteellisen vanhoja tutkimuksia. Uusia tutkimuksia on löydettävissä, mutta ne painottuvat hyvin kapea-alaisesti johonkin osa-alueeseen kuten tekoälyyn. Kirjallisuuskatsauksen ajantasaisuutta on kuitenkin pyritty vahvistamaan uusilla lähteillä, mikäli sellaisia on löydetty.

Kehittämisprojektin ensimmäisessä iteraatiossa haastateltiin kuutta ja toisessa iteraatiossa näistä neljää ohjelmistoprojektien kanssa työskentelevää henkilöä. Haas-

tateltavien määrä jäi vähäiseksi ja tämä vaikuttaa tulosten luotettavuuteen ja yleistettävyyteen. Suuremmalla haastateltavien määrällä olisi voitu ehkä saada enemmän tietoa esimerkiksi vihreään ohjelmistokehitykseen liittyviin aihepiireihin ja kyselyssä olevia aiheita ja kysymyksiä olisi ehkä voitu rajata enemmän tai vaihtoehtoisesti syventää tarvittavilta osin. Tulokset ovat siksi viitteellisiä ja tuotteesta olisi suurella todennäköisyydellä tullut erilainen isommalla haastateltavien määrällä. Tutkimuksen kirjallisuuskatsauksessa kerättiin yhteen keskeisimpiä vihreitä käytänteitä ja laadullisen tutkimuksen avulla pystyttiin todentamaan, että tietämys vihreistä käytänteistä on puutteellista, sitä voidaan lisätä informatiivisen kyselyn avulla ja työntekijät suhtautuvat positiivisesti vihreiden toimintatapojen adaptoimiseen ja ohjelmistojen vihreyden lisäämiseen. Pienen osallistujamäärän takia tuloksia ei voida kuitenkaan yleistää ja tarvittaisiin suurempi otanta, jotta tuloksista voitaisiin tehdä yleistettäviä johtopäätöksiä. Kehittämisen kohteena ollutta kontekstia on kuitenkin pyritty kuvaamaan siten, että tuotetta voidaan hyödyntää myös muissa kehittämisympäristöissä.

Ensimmäisen haastattelukierroksen tulokset olivat joiltakin osin ristiriitaisia. Esimerkiksi tuloksista selvisi, että kaikkia aihealueita pidettiin tärkeänä erityisesti siksi, että kaikkien ohjelmistokehitykseen osallistuvien olisi hyvä tietää vihreään ohjelmistokehitykseen liittyvistä asioista kokonaisvaltaisesti. Samalla kuitenkin pohdittiin sitä, että olisiko kysely aiheellista jakaa useammaksi kyselyksi ja näihin vastattaisiin työntekijän työnkuvan perusteella.

Ennen kehittämisprojektin aloittamista ennakko-oletuksena oli, että työntekijöillä on valmiiksi jonkinlainen mielikuva vihreästä ohjelmoinnista ja haastatteluilla olisi pystytty syventämään, lisäämään ja poistamaan vahvemmin kyselyssä esiintyviä aiheita. Yllätyksenä kuitenkin tuli, että vihreä ohjelmointi ja siihen liittyvät käytänteet olivat haastateltaville melko vieraita aiheita. Kaikki aiheet tuntuivat haastateltavista tärkeältä. Tämän takia kyselyn aiheiden rajaaminen oli haastavaa.

Ensimmäisten haastattelujen aikana eli alustavaa versiota arvioidessa haastateltavien oli vaikea hahmottaa kyselyn tarkoitusta ja informatiivista aspektia. Kysely tuntui tässä vaiheessa tenttimäiseltä ja jopa kovalta. Haastateltavat jopa sanoivat, että heille tulee tunne siitä, että arvostelu voisi vaikuttaa jopa töiden jatkumiseen organisaatiossa. Toisaalta tällä palautteella kyselyä pystyttiin kehittämään sellaiseksi, että se korostaa enemmän informatiivisuutta ja on luonteeltaan enemmän kannustava vihreiden käytänteiden käyttämiseen. Informatiivisuutta ja kannustavaa otetta pyrittiin jatkamaan myös kyselyn palautteessa. Kysely sai toisien haastattelujen ai-

kana enemmän positiivisempaa palautetta ja näistä tuli tunne, että kyselyä oli pystytty kehittämään tarkoituksenmukaisempaan suuntaan. Kaikestaan haastattelutilanteet olivat rentoja ja haastateltavat kertoivat avoimesti mielipiteensä kyselystä.

Tulosten perusteella kyselylomakkeen jatkokehittämisellä oli kolme suuntaa: ohjelmistokehitysprosessin osuus poistetaan kokonaan, ohjelmistokehitysprosessin osuudesta poistetaan kysymyksiä tai ohjelmistokehitysprosessin osuus jätetään sellaiseksi kuin se on. Ensimmäinen vaihtoehto eli ohjelmistokehitysprosessin osuuden poistaminen ei kuitenkaan olisi ollut tulosten mukainen, koska kyselylomakkeen kokonaisvaltaisuus sekä vihreyden mittaamiseen liittyvät asiat koettiin tärkeänä. Ohjelmistokehitysprosessin osuuden jättäminen ensimmäisen version kaltaiseksi ei kuitenkaan sekään ollut hyvä vaihtoehto, koska siinä oli kysymyksiä, jotka eivät vastanneet yrityksen käytänteitä. Tämän lisäksi kyselylomakkeen pituus koettiin joko pitkänä tai sopivan pituisena, joten muihin aihealueisiin tarvittavien lisäkysymysten lisääminen olisi pidentänyt kyselylomaketta edelleen ja tämä olisi ollut myös tulosten vastaista. Näiden syiden takia ohjelmistokehitysprosessin jättäminen kyselylomakkeeseen, mutta Scrum-prosessiin ja ketteriin menetelmiin liittyvien kysymysten poistaminen oli kompromissi ratkaisu tulosten ristiriitaisuuksiin.

Kyselylomakkeen pituuden rajaaminen oli haastavaa. Teemahaastattelun tuloksista pysty johtamaan joidenkin kysymysten poistamisen suoraan. Epäolennaisten kysymysten tai vastausvaihtoehtojen valitseminen jäi suurilta osin tutkimuksen tekijän päätettäväksi. Tässä apuna pyrittiin käyttämään kirjallisuuskatsausta yhdessä haastatteluaineiston kanssa. Kysymyksen poistamista puolsi se, jos kirjallisuuskatsauksesta ei löytynyt vahvaa perustetta kysymyksen säilyttämiselle. Toisaalta kysymyksen tai vastausvaihtoehdon säilyttämistä puolsi se, että kirjallisuuskatsauksesta löytyi perusteluita sen säilyttämiselle ja haastattelujen aikana aiheesta oli keskusteltu positiivisessa mielessä.

Olisi ollut myös mielekästä toteuttaa pyydetty tarkistuslista, jota työntekijät olisivat voineet käyttää työnteon tukena. Tämä olisi kuitenkin kasvattanut tutkimuksen työmäärää liiaksi. Toisaalta olisi ollut myös mielekästä kehittää aiheesta interaktiivinen minikurssi, johon olisi voinut yhdistää esimerkiksi harjoitustehtäviä kysymysten lisäksi. Tällaiseen kurssiin olisi voinut lisätä myös muuta visuaalisuutta kuvien ja videoiden muodossa, joka olisi palvellut audiovisuaalisia oppijoita paremmin.

Yleispätevän kyselyn laatiminen tästä aiheesta ja tällä tutkimusasetelmalla on mahdotonta. Tämä voidaan perustella sillä, että näinkin pienessä tutkimusasetel-

massa saatiin hyvin erilaisia näkemyksiä ja tulokset olivat joiltain osin ristiriitaisia. Myös jokaisen organisaation toimintakulttuuri on erilainen ja esimerkiksi ketteriä menetelmiä käyttävät yritykset tarvitsisivat ainakin ohjelmistokehitysprosessin osalta toisenlaisen kyselyn. Tästä voidaan tehdä johtopäätös, että tällaista kyselyä laadittaessa on perehdyttävä yritykseen ja heidän toimintatapoihinsa ja kehitettävä kysely, joka vastaa juuri kyseisen organisaation tarpeisiin. Kyselyn tulisi vastata myös organisaation strategisiin tavoitteisiin. Tästä huolimatta kyselyn voisi ajatella sopivan kaikille ohjelmistoprojektien kanssa työskenteleville. Kaikki kyselyssä esitetyt asiat eivät välttämättä sovellu sellaisenaan jokaisen organisaation käyttöön, mutta sitä voi kuitenkin soveltuvin osin hyödyntää. Kysely antaa laaja-alaisen mielikuvan ohjelmiston vihreyteen vaikuttavista asioista ja kannustaa ohjelmistoprojekteissa työskenteleviä omaksumaan vihreitä työtapoja ja ohjelmointikäytänteitä.

9 Yhteenveto

Tässä työssä kehitettiin vihreää ohjelmointia koskeva informatiivinen sähköinen kysely ja tutkittiin millaisella informatiivisella kyselyllä ohjelmistokehittäjien vihreiden ohjelmistokehitykseen liittyvien käytänteiden tietämystä voidaan mitata ja edistää. Kestävän kehityksen ja vihreiden arvojen tärkeys on korostunut nyky-yhteiskunnassa ja kaikkien alojen toimijoilla on tarve parantaa tuotteidensa vihreyttä ja vähentää hiilidioksidipäästöjä. Informatiivisen kyselyn hyödyntämisestä vihreän ohjelmointiosaamisen mittaamiseen ja parantamiseen on vain vähän tutkimusta ja siksi se oli mielekäs ja ajankohtainen tutkimusaihe. Tutkielma tehtiin yhteistyössä ICT-ratkaisuja ja -palveluja eri toimialoille toimittavalla Digia Oyj:n kanssa ja se toteutettiin Design Science Research Methology -prosessina.

Tutkielman tekeminen aloitettiin tutustumalla kestävän ohjelmiston määritelmiin ja laatutekijöihin. Tämän jälkeen siirryttiin vihreän ohjelmistokehitysprosessin menestystekijöihin, käytänteisiin, vihreyden mittaamiseen ja raportointiin. Ohjelmistokehitysprosessin jälkeen siirryttiin ohjelmiston elinkaarimallin vaiheiden vihreyteen vaikuttaviin asioihin ja ohjelmiston vihreyteen vaikuttaviin tekijöihin ja tekniikoihin. Lopuksi tehtiin katsausta mittareiden rakentamisesta.

Kehittämistyöhön liittyvä prosessi muodostui kahdesta iteraatiosta. Ensimmäisen iteraation aikana kysely rakennettiin muotoilemalla kirjallisuuskatsaukseen perustuvia kysymyksiä viisikymmentäkuusi kappaletta. Kysymysten muotoilemisen jälkeen käytiin palautekeskustelu Digia Oyj:n yhteyshenkilön kanssa. Kysely käytiin läpi yleisluontoisesti siten, että asiakokonaisuudet muodostivat tarkoituksenmukaisen kokonaisuuden. Tämän jälkeen toteutettiin teemahaastattelu kuudelle Digia Oyj:n työntekijälle. Haastatteluissa pyrittiin selvittämään haastateltavien mielipide kyselyn sisällöstä, ulkoasusta, kysymyksenasettelusta ja termistöstä. Sisällönanalyysin avulla pystyttiin tekemään johtopäätöksiä, joiden perusteella kyselyä kehitettiin edelleen ja siitä tehtiin sähköinen versio. Tämä mahdollisti ohjelmallisia toimintoja informatiivisuuden parantamiseksi. Toisessa iteraatiossa toteutettiin toinen suppeampi teemahaastattelu neljälle Digia Oyj:n työntekijälle. Myös tälle materiaalille toteutettiin sisällönanalyysi ja siitä tehtyjen johtopäätösten perusteella kyselyyn toteutettiin parannuksia. Viimeiseksi kysely demonstroitiin Digia Oyj:n yhteyshen-

kilölle, joka hyväksyi tuotteen.

Tässä tutkimuksessa kehitettiin informatiivinen kysely, joka koettiin olevan toimiva lähestymistapa vihreän koodaamisen käytänteiden opetteluun. Tuloksista kuitenkin selvisi, että kyselyn tehostamiseksi ennen kyselyä tulisi olla ennakkomateriaalia opiskeltavista asioista. Kyselyä pidettiin myös hankalana käytännön työhön vietynä, joten kyselyssä esitetyistä asioista tulisi olla saatavilla tarkistuslista. Tutkimuksessa onnistuttiin lisäämään tietoutta vihreästä ohjelmistokehityksestä ja luontainen jatkotutkimusaihe olisi se, että kuinka vihreät käytänteet ohjelmistokehityksessä voitaisiin vakiinnuttaa ohjelmistokehitysprojekteihin.

Kehitetty artefakti on tehty Digia Oyj:n yhteyshenkilön esittämiä tarpeita silmällä pitäen. Samalla se on myös haaste, koska olisi kuitenkin hyödyllistä, jos pystyttäisiin kehittää yleispätevä kysely, johon voitaisiin organisaatioiden tarpeiden mukaisesti lisätä osioita. Kyselystä voitaisiin myös jatkokehittää interaktiivinen minikurssi, johon voisi integroida opetuksellisia elementtejä.

Tutkimuksessa ei myöskään käsitelty tekoälyn roolia vihreän koodaamisen apuna ja tämä aihe itsessään sopisi tutkimuksen kohteeksi. Kysely on myös tarkoituksellisesti luonnoltaan enemmän informatiivinen, joten vihreän osaamisen mittaamisen kannalta kyselyn tulokset ovat suuntaa antavia. Osaamisen mittaamiseksi pitäisi kehittää juuri tähän tarkoitukseen oma mittarinsa.

Lähteet

[1] ABDULSALAM, S., LAKOMSKI, D., GU, Q., JIN, T., JA ZONG, Z. Program energy efficiency: The impact of language, compiler and implementation choices. Julkaisusarjassa *International Green Computing Conference* (2014), 1–6.

[2] ACHAR, S. Cloud computing: Toward sustainable processes and better environmental impact. *Journal of Computer Hardware Engineering (JCHE) 1*, 1 (2022).

[3] AGARWAL, S., NATH, A., JA CHOWDHURY, D. Sustainable approaches and good practices in green software engineering. *International Journal of Research and Reviews in Computer Science 3*, 1 (2012), 1425.

[4] AGGARWAL, K., HINDLE, A., JA STROULIA, E. Greenadvisor: A tool for analyzing the impact of software evolution on energy consumption. Julkaisusarjassa *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (2015), 311–320.

[5] AGGARWAL, K., ZHANG, C., CAMPBELL, J. C., HINDLE, A., JA STROULIA, E. The power of system call traces: predicting the software energy consumption impact of changes. Julkaisusarjassa *CASCON* (2014), vol. 14, 219–233.

[6] AHMAD IBRAHIM, S. R., YAHAYA, J., JA SALLEHUDIN, H. Green software process factors: A qualitative study. *Sustainability 14*, 18 (2022).

[7] BARAB, S., JA SQUIRE, K. Design-based research: Putting a stake in the ground. *The Journal Of The Learning Sciences 13*, 1 (2004), 1–14.

[8] CAPRA, E., FRANCALANCI, C., JA SLAUGHTER, S. A. Is software green? application development environments and energy efficiency in open source applications. *Information and Software Technology 54*, 1 (2012), 60–71.

[9] CHEN, P.-S. D., LAMBERT, A. D., JA GUIDRY, K. R. Engaging online learners: The impact of web-based learning technology on college student engagement. *Computers Education 54*, 4 (2010), 1222–1232.

[10] COLLECTIVE, D.-B. R. Design-based research: An emerging paradigm for educational inquiry. *Educational researcher 32*, 1 (2003), 5–8.

[11] DENY ARTHAWAN SUGIH, P. I. P., NUGROHO, E., JA HARTANTO, R. Analysis on green it applications usage for the firm's competitive advantage strategy. Julkaisusarjassa *2017 15th International Conference on Quality in Research (QiR) : International Symposium on Electrical and Computer Engineering* (2017), 29–33.

[12] DICK, M., DRANGMEISTER, J., KERN, E., JA NAUMANN, S. Green software engineering with agile methods. Julkaisusarjassa *2013 2nd International Workshop on Green and Sustainable Software (GREENS)* (2013), 78–85.

[13] DICK, M., NAUMANN, S., JA KUHN, N. A model and selected instances of green and sustainable software. Julkaisusarjassa *What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience: 9th IFIP TC 9 International Conference, HCC9 2010 and 1st IFIP TC 11 International Conference, CIP 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings* (2010), Springer, 248–259.

[14] DRESCH, ALINE, K., ANTUNES, JUNICO, K., JA LACERDA, DANIEL PACHECO, K. *Design science research : a method for science and technology advancement*. Springer, Cham, 2015.

[15] EDELSON, D. C. Design research: What we learn when we engage in design. *The Journal of the Learning sciences 11*, 1 (2002), 105–121.

[16] ENCARNACION, R. F. E., GALANG, A. A. D., JA HALLAR, B. J. A. The impact and effectiveness of e-learning on teaching and learning. *Online Submission 5*, 1 (2021), 383–397.

[17] ENGSTRÖM, E., STOREY, M.-A., RUNESON, P., HÖST, M., JA BALDASSARRE, M. T. How software engineering research aligns with design science: a review. *Empirical Software Engineering 25* (2020), 2630–2660.

[18] GEORGIOU, S., RIZOU, S., JA SPINELLIS, D. Software development lifecycle for energy efficiency: Techniques and tools. *ACM Comput. Surv. 52*, 4 (aug 2019).

[19] HANNAY, J. E., DYBÅ, T., ARISHOLM, E., JA SJØBERG, D. I. The effectiveness of pair programming: A meta-analysis. *Information and Software Technology 51*, 7 (2009), 1110–1122. Special Section: Software Engineering for Secure Systems.

[20] HANNULA, M., JA ANTTI, L. *Concetps of performance measurement, Suorituskyvyn mittauksen käsitteet*. Metalliteollisuuden kustannus Oy, 2002.

[21] HEVNER, A., R, A., MARCH, S., T, S., PARK, PARK, J., RAM, JA SUDHA. Design science in information systems research. *Management Information Systems Quarterly 28* (03 2004), 75–.

[22] HIRSJÄRVI, S., JA HURME, H. *Tutkimushaastattelu*. Gaudeamus Helsinkin University Press, 2014.

[23] HIRSJÄRVI, S., REMES, P., JA SAJAVAARA, P. *Tutki ja kirjoita*. Kustannusosakeyhtiö Tammi, 1996.

[24] IMPERATIVES, S. Report of the world commission on environment and development: Our common future. *Accessed Feb 10* (1987), 1–300.

[25] KAMBADUR, M., JA KIM, M. A. An experimental survey of energy management across the stack. *SIGPLAN Not. 49*, 10 (oct 2014), 329344.

[26] KANANEN, J. *Kehittämistutkimus opinnäytetyönä Kehittämistutkimuksen kirjoittamisen käytännön opas*. Jyväskylän ammattikorkeakoulu, Jyväskylä, 2012.

[27] KATAL, A., DAHIYA, S., JA CHOUDHURY, T. Energy efficiency in cloud computing data centers: a survey on software technologies. *Cluster Computing, The Journal of Networks, Software Tools and Applications* (2022).

[28] KATAL, A., DAHIYA, S., JA CHOUDHURY, T. Energy efficiency in cloud computing data centers: a survey on software technologies. *Cluster Computing 26*, 3 (2023), 1845–1875.

[29] KROSNICK, J. A., JA PRESSER, S. Question and questionnaire design.

[30] LANNELONGUE, L., GREALEY, J., JA INOUYE, M. Green algorithms: quantifying the carbon footprint of computation. *Advanced science 8*, 12 (2021), 2100707.

[31] LEE, H., CHOI, Y., VAN NGUYEN, T., HAI, Y., KIM, J., BAHJA, M., JA HOCAOĞLU, H. Covid19 led virtualization: Green data center for information systems research. *Information Systems Management 37*, 4 (2020), 272–276.

[32] LEE, J., SONG, H.-D., JA HONG, A. J. Exploring factors, and indicators for measuring students sustainable engagement in e-learning. *Sustainability 11*, 4 (2019).

[33] LINCOLN, Y. S., JA GUBA, E. G. *Naturalistic inquiry*. sage, 1985.

[34] LÖNNQVIST, A., KUJANSIVU, P., JA ANTOLA, J. *Aineettoman pääoman johtaminen*. JTO-Palvelut Oy, 2005.

[35] MALAVOLTA, I., CHINNAPPAN, K., JASMONTAS, L., GUPTA, S., JA ALI KARAM SOLTANY, K. Evaluating the impact of caching on the energy consumption and performance of progressive web apps.

[36] MANOTAS, I., BIRD, C., ZHANG, R., SHEPHERD, D., JASPAN, C., SADOWSKI, C., POLLOCK, L., JA CLAUSE, J. An empirical study of practitioners' perspectives on green software engineering. Julkaisusarjassa *Proceedings of the 38th International Conference on Software Engineering* (New York, NY, USA, 2016), ICSE '16, Association for Computing Machinery, 237248.

[37] MIRELES, G. A. G., MORAGA, M. Á., GARCÍA, F., JA PIATTINI, M. A classification approach of sustainability aware requirements methods. Julkaisusarjassa *2017 12th Iberian conference on information systems and technologies (CISTI)* (2017), IEEE, 1–6.

[38] MITRA, S., GUPTA, M., MISAILOVIC, S., JA BAGCHI, S. Phase-aware optimization in approximate computing. Julkaisusarjassa *CGO 2017 - Proceedings of the 2017 International Symposium on Code Generation and Optimization* (United States, 2017), V. Reddi, A. Smith, ja L. Tang, Eds., CGO 2017 - Proceedings of the 2017 International Symposium on Code Generation and Optimization, Institute of Electrical and Electronics Engineers Inc., 185–196.

[39] MOISES, A. C., MALUCELLI, A., JA REINEHR, S. Practices of energy consumption for sustainable software engineering. Julkaisusarjassa *2018 Ninth International Green and Sustainable Computing Conference (IGSC)* (2018), 1–6.

[40] MURUGESAN, S. Harnessing green it: Principles and practices. *IT Professional 10* (02 2008), 24 – 33.

[41] NOUREDDINE, A., JA RAJAN, A. Optimising energy consumption of design patterns. Julkaisusarjassa *New Ideas and Emerging Resuts (NIER) of the 37th International Conference on Software Engineering (ICSE'15). Florence, Italy, May 2015.* (May 2015).

[42] NUHFER, E., JA KNIPP, D. 4: The knowledge survey: A tool for all reasons. *To improve the academy 21*, 1 (2003), 59–78.

[43] PAUL, S. G., SAHA, A., AREFIN, M. S., BHUIYAN, T., BISWAS, A. A., REZA, A. W., ALOTAIBI, N. M., ALYAMI, S. A., JA MONI, M. A. A comprehensive review of green computing: Past, present, and future research. *IEEE Access 11* (2023), 87445–87494.

[44] PEFFERS, K., TUUNANEN, T., ROTHENBERGER, M. A., JA CHATTERJEE, S. A design science research methodology for information systems research. *Journal of management information systems 24*, 3 (2007), 45–77.

[45] PENZENSTADLER, B. Infusing green: Requirements engineering for green in and through software systems. Julkaisusarjassa *RE4SuSy@ RE* (2014), 44–53.

[46] PEREIRA, R., COUTO, M., RIBEIRO, F., RUA, R., CUNHA, J., FERNANDES, J. A. P., JA SARAIVA, J. A. Energy efficiency across programming languages: How do energy, time, and memory relate? SLE 2017, Association for Computing Machinery, 256267.

[47] PEREIRA, R., COUTO, M., RIBEIRO, F., RUA, R., CUNHA, J., FERNANDES, J. P., JA SARAIVA, J. Ranking programming languages by energy efficiency. *Science of Computer Programming 205* (2021), 102609.

[48] PERNAA, J. Kehittämistutkimus: Tieto-ja viestintätekniikkaa kemian opetukseen.

[49] PINTO, G., CASTOR, F., JA LIU, Y. D. Mining questions about software energy consumption. Julkaisusarjassa *Proceedings of the 11th Working Conference on Mining Software Repositories* (New York, NY, USA, 2014), MSR 2014, Association for Computing Machinery, 2231.

[50] RASHID, N., JA KHAN, S. U. Agile practices for global software development vendors in the development of green and sustainable software. *Journal of Software: Evolution and Process 30*, 10 (2018), e1964. e1964 JSME-17-0140.R3.

[51] SALAM, M., JA KHAN, S. U. Developing green and sustainable software: Success factors for vendors. Julkaisusarjassa *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)* (2016), IEEE, 1059–1062.

[52] SAPUTRI, T. R. D., JA LEE, S.-W. Integrated framework for incorporating sustainability design in software engineering life-cycle: An empirical study. *Information and Software Technology 129* (2021), 106407.

[53] SCRUM GUIDES. The 2020 scrum guide. URL `https://scrumguides.org/scrum-guide.html`, viitattu 22.5.2023.

[54] SHARMA, M., ARUNACHALAM, K., JA SHARMA, D. Analyzing the data center efficiency by using pue to make data centers more energy efficient by reducing the electrical consumption and exploring new strategies. *Procedia Computer Science 48* (2015), 142–148. International Conference on Computer, Communication and Convergence (ICCC 2015).

[55] STEIGERWALD, B., JA AGRAWAL, A. Developing green software. *Intel White Paper 9* (2011).

[56] TAINA, J. Good, bad, and beautiful software-in search of green software quality factors. *Cepis Upgrade 12*, 4 (2011), 22–27.

[57] THEUNISSEN, T., VAN HEESCH, U., JA AVGERIOU, P. A mapping study on documentation in continuous software development. *Information and Software Technology 142* (2022), 106733.

[58] TUOMI, J., JA SARAJÄRVI, A. *Laadullinen tutkimus ja sisällönanalyysi*. Kustannusosakeyhtiö Tammi, 2018.

[59] UUSI-RAUVA, E. *Ohjauksen tunnusluvut ja suoritusten mittaus*. Tampereen teknillinen korkeakoulu, 1994.

[60] VALLI, R., JA AALTOLA, J. Ikkunoita tutkimusmetodeihin: 1, metodin valinta ja aineistonkeruu: virikkeitä aloittelevalle tutkijalle. *Jyväskylä: PS-kustannus* (2018).

[61] VERDECCHIA, R., LAGO, P., EBERT, C., JA DE VRIES, C. Green it and green software. *IEEE Software 38*, 6 (2021), 7–15.

[62] VIITALA, R. *johda osaamista!: Osaamisen johtaminen teoriasta käytäntöön*. Inforviestintä Oy, 2005.

[63] WBCSD. Visio 2050. URL `https://fibsry.fi/wp-content/uploads/2022/03/Visio-2050_Muutoksen-aika_WBCSD.pdf`, viitattu 9.5.2023.

A Saatekirje haastateltaville

Tervetuloa osallistumaan vihreän ohjelmoinnin osaamisen kartoittamiseen tarkoitetun kyselylomakkeen kehittämistä koskevaan tutkimukseen! Tutkimus toteutetaan Jyväskylän yliopistossa vuoden 2023 aikana informaatioteknologian Pro Gradu -tutkielmaa varten.

Tutkimuksen tarkoituksena on kehittää teoriaan pohjautuva informatiivinen kyselylomake vihreän ohjelmointiosaamisen arvioimiseksi sekä kartoittaa tutkimukseen osallistuvien kokemuksia kyselylomakkeesta. Haastatteluiden tulosten avulla kyselylomaketta pyritään kehittämään ja täydentämään edelleen.

Haastattelut toteutetaan vähintään kahtena erillisenä teemahaastatteluna. Yhden haastattelun pituus on arvilta 30 – 50 minuuttia. Haastatteluun osallistuvat saavat liitteenä ohjelmistokehittäjien vihreän ohjelmoinnin osaamisen kartoittamisen kyselylomakkeen etukäteen tutustuttavaksi. Haastattelut toteutetaan etäyhteyden avulla ja haastattelut tallennetaan myöhempää käsittelyä varten. Haastattelutallenteita ja niiden pohjalta laadittuja litterointeja käsitellään luottamuksellisesti eikä niitä luovuteta kolmansille osapuolille tai säilytetä siten, että ne voisivat päätyä kolmansien osapuolten käsiin. Haastatteluja käytetään ainoastaan kyseessä olevaa tutkielmaa varten sekä sen sisältöä ja mahdollisia sitaatteja käytetään tutkielmassa siten, että haastateltavaa ei voida näiden perusteella tunnistaa.

Mikäli Teillä on kysyttävää tutkimuksen toteutusta tai tulosten käsittelyä koskien, voitte ottaa minuun yhteyttä joko sähköpostitse tai puhelimitse.

Kiitos etukäteen tutkimukseen osallistumisesta!

Ystävällisin terveisin,
Jenni Yrjänä
jenni.a.yrjana@jyu.fi
puh +358 40 8649 772

B Vihreän ohjelmoinnin osaamisen kartoittamisen kysely 1.0

Työtapoihin liittyvät kysymykset

Tässä osiossa kysytään vihreisiin työtapoihin liittyviä kysymyksiä.

1. Onko totta, että etätyöskentely vähentää hiilidioksidipäästöjä?

   Kyllä Ei

2. Onko totta, että töihin pyöräily tai yleisten kulkuvälineiden käyttäminen vähentää hiilidioksidipäästöjä?

   Kyllä Ei

3. Onko totta, että paperitulostaminen ei lisää merkittävästi hiilidioksidipäästöjä?

   Kyllä Ei

4. Onko totta, että tietokoneen sammuttaminen tai lepotilaan laittaminen lyhyiksi ajoiksi, esimerkiksi kahvi- tai ruokatauon ajaksi, ei merkittävästi vähennä tietokoneen energiankulutusta?

   Kyllä Ei

5. Onko totta, että etäyhteyksien avulla järjestettävät tapaamiset eivät vähennä hiilidioksidipäästöjä?

   Kyllä Ei

6. Onko kevyiden asiakaspäätteiden käyttämisellä työtehtävissä merkitystä hiilidioksidipäästöjen vähentämisessä?

   Kyllä Ei

Vihreään ohjelmistokehitysprosessiin liittyvät kysymykset

Tässä osiossa kysytään vihreään ohjelmistokehitysprosessiin liittyviä kysymyksiä.

1. Onko tarkistuslistojen käytöstä hyötyä vihreässä ohjelmistokehitysprosessissa?

   Kyllä Ei

2. Onko totta, että dokumentaation vähentäminen lisää ohjelmistokehitysprosessin ja ohjelmiston vihreyttä?

   Kyllä Ei

3. Onko totta, että jatkuvalla validoinnilla ei ole merkitystä ohjelmistokehitysprosessin vihreyden kannalta?

   Kyllä Ei

4. Onko totta, että tehokas viestintä on eräs vihreän ohjelmistokehitysprosessin menestystekijöistä?

   Kyllä Ei

5. Jos ohjelmistokehitystiimissä on vihreästä ohjelmistokehityksestä vastaava henkilö, mitkä ovat hänen vastuulla olevat asiat? Voit valita useita.

   (a) Vihreästä ohjelmistokehityksestä vastaava henkilö on vastuussa prosessin ja ohjelmiston vihreyden mittaamisesta.

   (b) Vihreästä ohjelmistokehityksestä vastaava henkilö ei ole vastuussa prosessin ja ohjelmiston vihreyden mittaamisesta, vaan vastuu on tiiminjohtajalla / Scrum Masterilla.

   (c) Vihreästä ohjelmistokehityksestä vastaava henkilö on vastuussa prosessin ja ohjelmiston vihreyden dokumentoimisesta.

   (d) Vihreästä ohjelmistokehitykestä vastaava henkilö ei ole vastuussa prosessin ja ohjelmiston vihreyden dokumentoinnista, vaan vastuu on jaettu tiimin kaikkien jäsenten kesken.

   (e) Vihreästä ohjelmistokehityksestä vastaava henkilö on vastuussa prosessin ja ohjelmiston vihreyden raportoinnista.

   (f) Vihreästä ohjelmistokehityksestä vastaava henkilö ei ole vastuussa prosessin ja ohjelmiston vihreyden raportoinnista, vaan vastuu on tiiminjohtajalla / Scrum Masterilla.

6. Väite: ohjelmistokehitysprosessin vihreyttä ei tarvitse arvoida koko prosessin ajan. Riittää, että prosessin vihreyttä arvioidaan kehittämisvaiheessa sekä testauksen aikana. Onko tämä totta?

   Kyllä Ei

7. Alla on kaksi väittämää. Valitse näistä se, joka tukee paremmin vihreää ohjelmistokehitysprosessia.

   (a) Asiakkaan mukana oleminen vihreässä ohjelmistokehitysprosessissa koko prosessin keston ajan on tarpeellista.

   (b) Asiakas on mukana vihreässä ohjelmistokehitysprosessissa vain vaatimusmäärittelyn aikana sekä ottamassa vastaan loppuraporttia.

8. Alla on sprinttiin / iteraatioon liittyviä väittämiä. Valitse näistä oikeat.

   (a) Sprintin kestolla ei ole vaikutusta ohjelmistokehitysprosessin vihreyteen.

   (b) Vihreän ohjelmistokehitysprosessin yksi menestystekijöistä on sprinttien / iteraatioiden pitäminen lyhyinä.

   (c) Sprintin loppuvaiheessa tulisi pitää vihreää ohjelmistokehitysprosessia ja ohjelmistoa koskeva arviointitapaaminen.

   (d) Sprintin / iteraation loppuvaiheessa ei tarvitse pitää vihreää ohjelmistokehitysprosessia ja ohjelmistoa koskevaa arviointitapaamista, vaan vihreyttä koskeva arviointi tapahtuu yhdessä sprintin / iteraation arvioinnin yhteydessä.

   (e) Sprintin / iteraation arvioinnin yhteydessä tulisi käydä läpi myös ohjelmistokehitysproessin ja ohjelmiston vihreyttä koskevat tulokset.

9. Onko totta, että prosessin vihreyttä dokumentoiva päiväkirja on osa vihreää ohjelmistokehitysprosessia?

   Kyllä Ei

10. Onko totta, että ohjelmistokehitysprosessin päättyessä pidettävä ohjelmiston vihreyttä koskeva retrospektiivi on hyvä tapa lisätä tulevien projektien vihreyttä?

    Kyllä Ei

Vihreän ohjelmiston ja ohjelmistokehitysprosessin mittaaminen

1. Onko totta, että vihreän ohjelmistokehitysprosessin tärkein mittari on hiilijalanjäljen mittaaminen?

   Kyllä Ei

2. Onko totta, että ohjelmistokehitysprosessin aikaisen / tuottaman hukan määrää voi myös mitata?

   Kyllä Ei

3. Onko totta, että ohjelmistokehitysprosessin aikaista matkustamisen määrään on syytä seurata?

   Kyllä Ei

Vihreän ohjelmiston elinkaarimalli

Tässä osiossa kysytään vihreän ohjelmiston elinkaareen liittyviä kysymyksiä.

Vaatimusmäärittelyvaihe

1. Onko totta, että ohjelmiston vihreyttä koskevat vaatimukset tulisi asettaa tarkoilla määritelmillä?

   Kyllä Ei

2. Onko totta, että energiankulutusta koskevat vaatimukset tulisi ilmaista tarkoilla määritelmillä?

   Kyllä Ei

3. Onko totta, että epätarkat ja vaillinaiset vaatimukset voivat johtaa esimerkiksi siihen, että joutokäynnin aikainen energiankulutus huomioidaan paremmin kuin suorituksenaikainen energiankulutus?

   Kyllä Ei

4. Onko totta, että ohjelmistolle asetetut vaatimukset voivat olla ristiriidassa keskenään ja johtaa esimerkiksi siihen, että taloudelliset vaatimukset asetetaan ohjelmiston vihreyttä koskevien vaatimusten edelle?

   Kyllä Ei

5. Mikä on vihreä arvioija? Valitse oikea.

   (a) Ohjelmisto, joka arvioi kehitettävän ohjelmiston energiankulutusarvion.

(b) Ohjelmisto, joka arvioi ohjelmistokehitysprosessin energiankulutusarvion.

(c) Ohjelmisto, joka arvioi kehitettävän ohjelmiston hiilijalanjäljen.

Suunnitteluvaihe

1. Onko totta, että säännöllisten suunnittelukokousten pitäminen on yksi vihreän ohjelmiston menestystekijöistä?

   Kyllä Ei

2. Onko totta, että vihreän ohjelmistosuunnitelun näkökulmasta, ohjelmistoja suunniteltaessa ei tarvitse ottaa huomioon vanhempia laitteistoja?

   Kyllä Ei

3. Alla on suunnitteluvaiheeseen liittyviä väittämiä. Valitse näistä oikeat.

   (a) Ohjelmistot tulisi suunnitella siten, että ne eivät kuluta ainakaan enempää energiaa kuin edellinen versio.

   (b) Ohjelmistoja suunniteltaessa ei tarvitse huomioida edellisen version energiankulutusta.

   (c) Ohjelmistot tulisi suunnitella siten, että ne eivät kuluttaisi ainakaan enempää muistia kuin edellinen versio.

   (d) Ohjelmistoja suunniteltaessa ei tarvitse huomioda edellisen version muistinkäyttöä.

   (e) Ohjelmistot tulisi suunnitella siten, että ne eivät kuluttaisi ainakaan enempää kaistanleveyttä kuin edellinen versio.

   (f) Ohjelmistoja suunniteltaessa ei tarvitse huomioida edellisen version kaistanleveyden kulutusta.

4. IoT-laitteiden suunnittelussa ja toteutuksessa on paljon hyviä resursseja säästäviä periaatteita, joita voidaan hyödyntää myös muussa ohjelmoinnissa. Valitse näistä oikeat.

   (a) Ohjelmistot suunnitellaan siten, että ne eivät kulutua virtaa jatkuvasti.

   (b) Ohjelmistot suunnitellaan siten, että ne eivät käytä jatkuvaa tietoliikenneyhteyttä.

   (c) Ohjelmistot pyritään suunnittelemaan siten, että ne välttävät ylimääräistä muistinkäyttöä.

(d) Ohjelmistot pyritään suunnittelmaan siten, että ne käyttävät mahdollisimman tehokkaita algoritmeja.

Kehittämisvaihe

1. Valitse oikeat väitteet koskien pariohjelmointia.

   (a) Pariohjelmointi voi vähentää yksinkertaisten tehtävien suorittamiseen käytettävää aikaa.

   (b) Pariohjelmointi voi lisätä yksinkertaisten tehtävien suorittamiseen käytettävää aikaa.

   (c) Pariohjelmointi helpottaa monimutkaisten tehtävien ratkaisemista.

   (d) Pariohjelmointi vaikeuttaa monimutkaisten tehtävien ratkaisemista.

   (e) Pariohjelmoinnilla voidaan ratkaista monimutkaisia tehtäviä laadukkaammin.

   (f) Pariohjelmoinnilla monimutkaisten tehtävien ratkaisut ovat usein huonolaatuisia.

2. Voidaanko uudelleenkäytettävällä koodilla vähentää energiankulutusta ja hiilidioksidipäästöjä?

   (a) Kyllä

   (b) Ei

3. Valitse oikeat väitteet koskien komponenttipohjaista kehitysstrategiaa.

   (a) Komponenttipohjainen kehitysstrategia vähentää ajankäyttöä ja kustannuksia prosessissa.

   (b) Komponenttipohjainen kehitysstrategia lisää ajankäyttöä ja kustannuksia prosessissa.

   (c) Komponenttipohjainen kehitysstrategia mahdollistaa ohjelmiston osien uudelleenkäytön.

   (d) Komponenttipohjainen kehitysstrategia estää ohjelmiston osien uudelleenkäytön.

   (e) Komponenttipohjaisen ohjelmiston refaktoroinnon avulla voidaan selkeyttää ja tehostaa ohjelmistoa.

(f) Komponenttipohjaisen ohjelmiston tehostaminen refaktoroinnin avulla on mahdotonta.

Testausvaihe

1. Onko totta, että testitapausten suorittaminen testausstrategian mukaisesti vähentää energiankulutusta ja parantaa testauksen laatua?

   Kyllä Ei

2. Onko totta, että jatkuvan integraatioympäristön testipakettiin ei voi sisällyttää energia- ja suorituskykymittauksia?

   Kyllä Ei

3. Väite: Raakaan voimaan perustuva testaus korreloi suoraan energiankulutuksen kanssa.

   Kyllä Ei

4. Alla on lueteltu väitteitä, joiden mukaan testausta voidaan tehostaa. Valitse näistä oikeat.

   (a) Staattisen koodianalyysin avulla voidaan tehostaa testausta.

   (b) Testitapaukset ja testien hallinta voidaan instrumentoida testien tehokkuuden tunnistamiseksi.

   (c) Testattavuutta voidaan parantaa testivetoisella kehityksellä sekä testilähtöisen vaatimussuunnittelun avulla.

   (d) Testivetoisella kehityksellä sekä testilähtöisen vaatimussuunnittelun avulla voidaan helpottaa regressiotestausta ja tehdä siitä tehokkaampaa.

   (e) Testauksen tehokkuutta voidaan parantaa kohdistamalla testausta aiemmin testaamattomiin koodisegmentteihin.

   (f) Tekoäly voi auttaa määrittelemään mitkä testitapaukset on toteutettava ja missä määrin ja siten tehostaa testausta.

   (g) Älykäs validointi voi auttaa testitapausten valinnassa ja luomisessa ja siten tehostaa testausta.

Ylläpitovaihe

1. Onko totta, että ylläpitovaiheen vihreyteen liittyvät asiat tulisi ottaa huomioon jo tuotteen suunnittelu- ja kehittämisvaiheessa?

   Kyllä Ei

2. Onko totta, että ylläpitovaihe ei ole merkittävässä osassa ohjelmiston vihreyttä arvioitaessa?

   Kyllä Ei

3. Onko totta, että ylläpitovaiheen vihreyttä voidaan kasvattaa tarjoamalla käyttäjille energiansäästöohjeita sekä tapoja säätää ohjelmiston toimintaa energiaystävällisemmäksi?

   Kyllä Ei

Vihreään ohjelmistoon liittyvät kysymykset

Ohjelmiston vihreyden mittaaminen

1. Onko totta, että vihreä ohjelmisto säästää resursseja ja vähentää hukkaa?

   Kyllä Ei

2. Alla on listattu asioita, joilla mitataan ohjelmiston tehokkuutta. Valitse näistä oikeat.

   (a) CPU-intensiteetti

   (b) Muistin käyttö

   (c) Perifeerinen intensiteetti

   (d) Joutokäynti

   (e) Heijastuskyky

3. Onko totta, että kahden ohjelmiston arvoa kestävässä kehityksessä ei voi verrata, jos ne eivät ole samanlaisessa ohjelmistojärjestelmässä?

   Kyllä Ei

4. Onko totta, että kestävä kehitys voidaan määritellä tarkoitukseen sopivuudella, reduktiolla ja kauneudella?

   Kyllä Ei

Säästävät ohjelmistostrategiat

1. Onko totta, että säästävillä ohjelmistostrategioilla pyritään minimoimaan CPU:n ja muistin käyttöä?

   Kyllä Ei

Pilvipalvelut ja reunalaskenta

1. Onko totta, että pilvipalvelujen käyttäminen lisää ohjelmiston energiankäyttöä?

   Kyllä Ei

2. Alla on listattu väitteitä siitä, miksi pilvipalvelut ovat hyödyllisiä ajateltaessa vihreää ohjelmistokehitystä. Valitse näistä oikeat.

   (a) Pilvipohjaisissa palveluissa resursseja käytetään pyynnöstä, joka mahdollistaa isossa mittaskaalassa datakeskusten energiatehokkuuden optimoinnin.

   (b) Pilvipohjaisissa palveluissa resursseja käytetään pyynnöstä, joka mahdollistaa asiakaslaitteiden energiatehokkuuden optimoinnin.

   (c) Pilvipalveluissa on vähemmän yleiskustannuksia sekä tehokkaampi skaalautuvuus.

   (d) Pilvipalveluissa on enemmän yleiskustannuksia, mutta tehokkaampi skaalautuvuus.

   (e) Datakeskukset voivat tehokkaammin hyödyntää vihreitä energianlähteitä ja laitteiden lämpenemisestä syntyvän hukkaenergian.

   (f) Laitteiden joutokäyntiä on vähemmän.

3. Pelkästään pilvipalvelujen käyttäminen ei takaa kuitenkaan vihreämpää ohjelmistoa. Mitä ohjelmistokehittäjän tulee ottaa huomioon, jotta pilvipalvelujen käyttämisestä saadaan oletettu hyöty? Valitse oikeat.

   (a) Ohjelmistoarkkitehtuuri tulee suunnitella pilvipalvelujen käyttöä ajatellen.

   (b) Datan siirtely tulee optimoida.

   (c) Kaksoisdata tulee pyrkiä poistamaan.

   (d) Ohjelmistokehittäjän tulisi hyödyntää älykästä tietojen pakkaamista datan käyttötiheyteen perusteuen.

(e) Ohjelmiston tulisi hyödyntää myös palvelitonta laskentaa.

(f) Ohjelmiston tulisi hyödyntää Faas-toimintoa (function-as-service).

(g) Ohjelmistossa tulisi tarpeen mukaan hyödyntää datan käytön profilointia.

Ohjelmointikieli

1. Onko totta, että ajallisesti nopein ohjelmointikieli on myös tehokkain?

   Kyllä Ei

2. Onko totta, että tulkitut kielet ovat energiatehokkaampia muihin kieliin verrattuna?

   Kyllä Ei

3. Onko totta, että on mahdollista valita yksi ohjelmointikieli, joka on parempi muihin ohjelmointikieliin verrattuna?

   Kyllä Ei

4. Onko totta, että C-pohjaiset kielet eivät ole kovinkaan energiatehokkaita?

   Kyllä Ei

Datan ja muistin käyttö

1. Onko totta, että kokonaismuistin määrän ja DRAM-muistin energiankulutuksen välillä on suora yhteys?

   Kyllä Ei

Sähköinen jäte

1. Onko totta, että sähköisellä jätteellä voidaan tarkoittaa elektroniikkajätettä tai datajätettä?

   Kyllä Ei

2. Onko totta, että sähköinen jäte ei ole merkittävä tekijä ohjelmiston hiilijalanjälkeä arvioitaessa?

   Kyllä Ei

3. Onko totta, että sähköistä jätettä syntyy vaatimusmäärittelyvaiheessa esimerkiksi silloin, kun vaatimukset ovat kirjattu virheellisesti ja ne eivät vastaa asiakkaan vaatimuksia?

   Kyllä Ei

4. Onko totta, että monimutkainen koodi ja dokumentaatio lisäävät sähköistä jätettä?

   Kyllä Ei

5. Onko totta, että ohjelmoijan puutteelliset taidot lisäävät sähköistä jätettä?

   Kyllä Ei