KASVATUS & AIKA

KATSAUS

https://doi.org/10.33350/ka.111888

Tietokonevallankumous ja ohjelmoinnillinen ajattelu peruskoulussa – Havaintoja mikro- ja makrotasoilta

Janne Fagerlund

Lectio praecursoria kasvatustieteen väitöskirjaan Teaching, Learning and Assessing Computational Thinking through Programming with Scratch in Primary Schools, Jyväskylän yliopistossa 5.11.2021.

Tietokonevallankumous on täällä

Elämme tietokonevallankumouksen aikakautta. Merkittävä osa nykyisin käyttämistämme laitteista sekä palveluista toimii tietokoneiden ja niiden suorittamien tietokoneohjelmien kautta. (Denning & Tedre 2019.) Arvelen, että jokaisella meistä on mukanamme vähintään yksi tietokone aina kun poistumme kotoamme. Se voi olla kännykässämme, joka kenties seuraa sijaintiamme, on oppinut puheäänemme ja tietää suosikkisovelluksemme. Tiesittekö, että nuo puhelimet ovat itse asiassa voimakkaampia kuin tietokone, joka lähetti Neil Armstrongin kuuhun vuonna 1969? Oletettavasti tietokoneita ja tietokoneohjelmia on myös älykelloissamme ja urheilurannekkeissamme, jotka keräävät meistä tietoa. Ajelemme autoilla, joissa on ajotietokone – mutta totta puhuen noihin autoihin kätkeytyy useita tietokoneita, joista koko auton toiminta on riippuvainen. Kannamme usein laukuissamme pientä kannettavaa tietokonetta, jolla voi tehdä uskomattoman monimutkaisia ammattityötehtäviä ja vapaa-ajan toimia.

Tietokoneet ja erilaisten tietokoneohjelmien luonti ovat tuottaneet valtavia hyötyjä, kuten saaneet ihmisiä yhteen virtuaalisesti poikkeusaikana ja mahdollistaneet tartuntatautien parannuskeinojen kehittämistä algoritmisesti.

Tietokoneet ja tietokoneohjelmat ovat toki tuoneet maailmoihimme myös merkittäviä huolenaiheita (Denning & Tedre 2019). Kenties olemme isossa kuvassa huolissamme kybersodankäynnin muodoista, kuten palvelunestohyökkäyksistä tai propagandaksikin kai laskettavista valeuutisista Internetissä. Ehkä automaatio on muuttanut työnkuvaamme radikaalisti. Saatamme olla huolissamme, ketkä voivat seurata tarkkaa sijaintiamme mobiililaitteistamme, tai kenellä on pääsy kulutustottumuksiimme tai mieltymyksiimme, joista jättiyhtiöt räätälöivät meille yksilöllistä mainontaa maksimoidakseen mainostulojaan. Olemme mahdollisesti huomanneet, että sosiaalisen median algoritmit ovat joskus johdattaneet meidät informaatiokuplaan, eli törmäämme toistuvasti yksipuoliseen sisältöön, kuten rikkaiden

ja menestyneiden ihmisten somepäivityksiin. Jonkun laatima tietokoneohjelma se sielläkin on ollut ohjaamassa palvelun käyttöämme jollakin tarkoitusperällä.

Ohjelmoinnillisen ajattelun tulo kouluihin

Koulun ydintehtävänä on kehittää oppilaiden perusymmärrystä maailmasta ja opettaa elämisen perustaitoja. Voimme varmasti tulla melko yksimielisesti siihen päätelmään, että koulu ei saa unohtaa tällaisten yhteiskuntaa ravisuttavien ohjelmoinnillisten ilmiöiden huomioimista. Voisi sanoa, että joitakin ohjelmoinnin ilmiöitä on varmasti tärkeä oppia jo peruskoulussa, ja myös opettajien tulee olla valmiita ottamaan aihealuetta jotenkin mukaan opetukseensa. (esim. Lonka ym. 2018; Mertala ym. 2020.) Oleellisempi kysymys ehkä kuuluukin: mitä ohjelmoinnillisuudesta olisi hyvä oppia ja miten sitä voitaisiin mielekkäästi opettaa? Kuten esimerkiksi matematiikka pyrkii kehittämään oppilaiden matemaattista ajattelua ja kuvataide taiteellista ajattelua, myös ohjelmoituun maailmaan on lanseerattu oma ydinosaamisalueensa: ohjelmoinnillisen ajattelun taito (Wing 2006).

Tämä monille uusi osaamisalue, ohjelmoinnillinen ajattelu, tuli kouluihimme uuden opetussuunnitelman myötä noin viisi vuotta sitten. Tietokoneohjelmien itse tekeminen eli ohjelmointi, kansankielisesti koodaaminen, tuli kaikille oppilaille pakolliseksi oppisisällöksi. (Opetushallitus 2014.) Samaa trendiä ovat noudattaneet myös monet muut maat (Heintz ym. 2016). Lapsille ja nuorille mielekkäiksi osoittautuneet koodaamisen tavat, kuten robottien ohjelmoiminen tai omien pelien, animaatioiden ja keksintöjen tekeminen ovat yleistyneet (Falkner ym. 2019; Hsu ym. 2018). Tällaisten toimien kautta oppilaille halutaan opettaa jotain tärkeää ohjelmoidusta todellisuudesta. Heistä pyritään tekemään ohjelmoinnillisia tekijöitä ja sen ymmärtäjiä, eikä vain ohjelmoitujen asioiden passiivisia kuluttajia (Grover & Pea 2013).

Sain itse mainion tilaisuuden tutkia tätä kouluillemme uutta ja ajankohtaista aihepiiriä. Tulin tutkimuksen äärelle sopivaan aikaan ponnistaen opettajankoulutustaustastani sekä harrastuneisuudestani teknologiaan. Sain tietää, että uutta tutkimusta tarvittiin kuumeisesti, sillä aihepiiri oli vähän tutkittu ja haastava opettajille. Tuoreiden tutkimusten mukaan opettajat eivät koe osaavansa opettaa ohjelmointia kovin hyvin, ja oppilaiden ohjelmoinnin osaaminen on verrattain matalaa (Fraillon ym. 2019; Kaarakainen ym. 2017; Leino ym. 2019).

Scratch ohjelmoinnillisen ajattelun perusperiaatteiden opetuksessa

Väitöskirjani keskiössä oli ohjelmointiympäristö Scratch. Scratch on Yhdysvalloissa kehitetty, hyvin suosittu, ilmainen, verkkoselaimella käytettävä ympäristö, jossa lapset ja nuoret voivat ohjelmoida omaperäisiä tietokoneruudulla näkyviä interaktiivisia tarinoita tai vaikkapa pelejä. Omiin ohjelmointitöihin valitaan sopivia hahmoja ja taustoja, kuten vaikkapa taruolentoja fantasiamaailmassa. Oman työn luomuksille voidaan koodata erilaisia toimintoja, kuten animaatioita tai repliikkejä, käyttämällä aloittelijoille suunnattua värikästä kuvakepohjaista ohjelmointikieltä. (Maloney ym. 2010.)

Scratch on yksi ohjelmointiympäristö muiden sellaisten muassa, joiden kautta oppilaiden on määrä oppia ohjelmoinnillista ajattelua (Brennan & Resnick 2012). Aihepiiri on tosin ollut kaikkea muuta kuin täysin selkeä. Tutkijat ja opettajat ovat pähkäilleet muun muassa seuraavia kysymyksiä: Millaista ohjelmoinnin kautta opittava laajempi osaaminen, ohjelmoinnillinen ajattelu, pohjimmiltaan on? Kuinka paljon sitä täytyy oppia? Miten sen

oppimista tuetaan? (Fagerlund ym. 2021.) Valmiita vastauksia on vähän, ja itse asiassa eri näkemykset ovat osittain kiisteleviäkin.

Luulen ymmärtäväni, miksi ohjelmoinnin opetus on ollut ongelmallista. Se johtuu ainakin siitä, että aihepiiri on poikkitieteellinen: se sijoittuu kahden tieteenalan, kasvatustieteen ja tietojenkäsittelytieteen, oppiainedidaktiseen välimaastoon (Denning & Tedre 2019). Tällä yhdistelmällä ei ole peruskoulussa vahvoja juuria, eikä se näin ollen ole perinteisesti kuulunut opettajien koulutukseenkaan. Tietotekniikkaa on toki ollut peruskouluissa pitkään, mutta se on eri asia kuin tietojenkäsittelytiede, joka on ikään kuin ohjelmoinnillisen ajattelun oppiaineellinen juurakko. Nyt tietokonevallankumous ja koodaamisen tulo kouluihin on tuonut insinöörit ja tietojenkäsittelytieteilijät vahvemmin yhteen kasvatusalan asiantuntijoiden kanssa. Yhteisissä ponnisteluissa on pyritty kirkastamaan niinkin perustavanlaatuisia asioita kuin sanastoa, jota ohjelmoinnillisen ajattelun opetuksessa sovelletaan (Barr & Stephenson 2011). Ennen kaikkea on pyritty kirkastamaan, mitä oppilaiden on tärkeä oppia. Ruokimmeko heidän koodaamistaitojaan? Vai opetammeko jonkinlaisia yleismaailmallisia ongelmanratkaisutaitoja, joita voidaan oppia kenties koskemattakaan tietokoneeseen? (Ks. esim. Denning 2017.)

Omassa väitöskirjassani näen asian seuraavalla tavalla. Jokaisen oppilaan olisi tärkeä oppia ymmärtämään, mitä ohjelmoinnilla voidaan ylipäänsä tehdä, miten ohjelmoidut asiat maailmassa toimivat ja miten ohjelmointia voidaan soveltaa käytännössä oikean elämän ongelmien ratkaisemisessa. Näiden ajatusten konkretisoimiseksi hahmottelin tutkimukseeni ohjelmoinnillisen ajattelun perusprinsiipit – keskeiset ohjelmoinnilliset asiat, joita lasten ja nuorten olisi hyvä omaksua tullakseen taitaviksi ohjelmoinnillisiksi ajattelijoiksi. (Fagerlund ym. 2021.)

On tärkeää kuitenkin korostaa, etteivät kaikki jaa tätä näkemystä. Eräskin tutkimusartikkelini vertaisarvioija jostain päin maailmaa ilmaisi, että ohjelmointia ei pitäisi opettaa koulussa. Koenkin pitkälti liikkuneeni vellovalla tieteellisellä maaperällä, jossa joutuu väistämättä etsimään uusia ratkaisuja sen sijaan, että turvautuisi hyväksi todettuihin teorioihin. Mutta tästähän tieteellisessä tutkimuksessa onkin kyse—uuden etsimisestä ja avaamisesta; joskin samalla oman ajattelukyvyn ja varmuudentunteen koettelusta.

Tutkimuksesta eväitä opetussuunnitelmaan ja luokkahuoneisiin

Väitöskirjatutkimukseni huomio kiinnittyi pääasiassa siihen, miten oppilaat oppivat ohjelmoinnillista ajattelua ala-asteella Scratch-ohjelmointiympäristössä ja miten heidän oppimistaan voitaisiin mielekkäästi tukea.

Aloitin toteuttamalla katsauksen (Fagerlund ym. 2021) aiempaan tutkimuskirjallisuuteen. Halusin tietää, miten hahmottelemani ohjelmoinnillisen ajattelun perusprinsiipit voitaisiin tulkata lasten ja nuorten kielelle Scratchissa. Koostin kokonaiskäsityksen siitä, ”mitä” asioita oppilaat voivat ohjelmoida Scratchissa sekä sitä, ”miten” he voivat toteuttaa ohjelmointia Scratchissa tavoilla, jotka jotenkin ruokkivat heidän laaja-alaisempaa osaamistaan, ohjelmoinnillista ajatteluaan. Tämä katsaus toimi teoreettisena peruskivenä tutkimukseni toiselle osiolle, joka oli luonteeltaan empiirinen.

Toteutin koulukokeilun eräässä peruskoulussa. Pidin 12 oppituntia ohjelmoinnin opetusta kolmelle neljäsluokalle, yhteensä 57 oppilaalle. Teimme erilaisia luovia Scratch-ohjelmointiharjoitustöitä, kuten omia animaatioita ja pienoispelejä. Jakson lopuksi oppilaat saivat keksiä ja toteuttaa omavalintaisen laajemman ohjelmointityön. Oppilaat työskentelivät jakson aikana pääsääntöisesti pareittain yhteisen tietokoneen äärellä.

 https://doi.org/10.33350/ka.111888

Keräsin oppilaiden työskentelystä kaksi pääasiallista aineistokokoelmaa: oppilaiden ohjelmoimat moninaiset Scratch-ohjelmointityöt sekä videotallenteet neljän oppilasparin pariohjelmoinnista vapaavalintaisten ohjelmointitöidensä aikana. Oppilaiden ohjelmointitöistä tutkin, mitä niihin koodatut ohjelmointisisällöt kielivät heidän ohjelmoinnillisen ajattelun taidoistaan (Fagerlund ym. 2020). Oppilasparien ohjelmointityöskentelyvideoista tutkin, millaisilla tavoilla parit toteuttivat erilaisia ohjelmoinnillisen ajattelun käytänteitä, kuten yhteistyöskentelyä ja ohjelmointivirheiden korjaamista eli debuggaamista (Fagerlund ym., arvioinnissa). Kaikkiaan tavoitteenani oli hankkia lisää rikasta käytännön tutkimustietoa koululaisten oppimisesta ja oppia paremmin ymmärtämään, miten oppimista voitaisiin luokkahuoneissa tehostaa.

Tutkimukseni tulokset olivat monipuolisia, ja niistä voidaan ottaa oppia moneen tärkeään kasvatusalan teemaan. Ensinnäkin tutkimukseni tuloksia voidaan peilata makrotasolla valtakunnalliseen perusopetuksen opetussuunnitelmaan (Opetushallitus 2014), jossa ohjelmoinnin oppimäärä on nykymuodossaan melko suppea. Tutkimukseni mukaan ohjelmoinnillinen ajattelu voidaan nähdä monipuolisena tieto- ja taitokokonaisuutena. Sitä voidaan ruokkia tutustumalla ohjelmoituun maailmaan ja ohjelmoimalla koulussa eri tavoin ja huomattavasti useammin kuin kertakokeiluluontoisesti. Tämän taitoalueen ainutlaatuinen ominaisluonne edellyttänee, että sitä opetetaan määrätietoisesti esimerkiksi sisällyttämällä ohjelmoinnillisen ajattelun perusprinsiipejä opetussuunnitelmaan ja osaksi opettajien osaamista.

Havainnoistani voidaan saada lisäksi mikrotasolla oppia ohjelmoinnillisen ajattelun jalkauttamiseen kouluopetukseen. Hahmotan tätä kolmen formatiiviseen arviointiin (Black & Wiliam 2009) liittyvän näkökulman kautta.

Ensimmäinen näkökulma, johon tutkimuksestani voidaan saada oppia ikään kuin ruohonjuuritasolla, liittyy oppimistavoitteisiin (Black & Wiliam 2009): mitä oppilaiden halutaan oppivan vaikkapa kullakin eri oppitunnilla tai eri ohjelmointitöissä? Ohjelmoinnillisen ajattelun perusoppisisältö eli sen substanssi on ollut aiemmin jokseenkin yleisluontoisesti ja ehkä hankalatajuisestikin kuvattu (Fagerlund ym. 2021). Tutkimukseni johdosta tämän aihealueen aakkoset ovat nyt konkreettisemmat erityisesti Scratch-ohjelmoinnin kontekstissa, ja niitä voidaan nyt kuvata jopa karkeana oppimisen polkuna alkeista eteenpäin. Tutkimuksessani konkretisoituja oppisisältöjä voidaan poimia oppimisen tavoitteiksi erilaisiin ohjelmointitöihin vaikkapa jokaiselle oppilaalle yksilöllisesti kunkin omaan tilanteeseen sopivasti.

Toinen näkökulma, johon tutkimuksestani voidaan saada oppia ruohonjuuritasolla, liittyy oppimisen arviointiin (Black & Wiliam 2009): mistä tiedämme, kuinka hyvin oppilaat ovat oppineet? Tutkimuksessani arvioin sekä oppilaiden ohjelmointitöitä että heidän ohjelmointikäytänteitään (Grover ym. 2017). Pidän perinteisiä kokeita jokseenkin epätarkoituksenmukaisina tässä aihepiirissä. Sekä oppilaiden oikeat ohjelmointityöt että heidän kulloisetkin tapansa ohjelmoida kielivät hyvin heidän osaamisestaan. Ne kielivät siitä, kuinka hyvin he osaavat soveltaa hankkimiaan ohjelmoinnillisen ajattelun taitoja käytännössä juuri sinä hetkenä. Sekä oppilaiden ohjelmointitöitä että heidän tapojaan ohjelmoida voi siis olla hedelmällistä tarkkailla nimenomaan arvioivin silmin.

Kolmas näkökulma, johon tutkimuksestani voidaan saada oppia ruohonjuuritasolla, liittyy palautteen antamiseen (Black & Wiliam 2009): miten voimme auttaa oppilaiden oppimista, kun he ohjelmoivat? Tutkimukseni vahvisti käsitystä siitä, että oppilaille on hyvä jättää tilaa itse löytämiselle ja oivaltamiselle: kyllähän esimerkiksi tietokone koko ajan kertoo, toimiiko ohjelma eli onnistuiko ohjelmointi vai ei (Ben-Ari 1998; Brennan & Resnick 2012). Tästä huolimatta oppilaat voivat usein tarvita ihmisen apua. Heille voikin olla hyvä

päästä työskentelemään pareittain, kysymään apua muilta oppilaita ja hakemaan tietoa Internetistä. Tällaisten käytänteiden toteuttaminen voi kuitenkin olla erityisesti pienemmille oppilaille haastavaa, joten opettajien on tarve tukea työskentelyä. Edellytyksenä voi puolestaan olla opettajan oma riittävä osaaminen ohjelmoinnillisessa ajattelussa.

Kasvatusalan toimijoiden kiinnittäminen aihepiiriin tärkeää

Väitöskirjani nojalla totean, että jokaisella oppilaalla tulisi olla oikeus oppia ohjelmointia ja ohjelmoinnillista ajattelua aina esiopetuksesta korkeakoulutukseen. Olen ollut sikäli onnekas tutkija, että olen saanut runsaasti tilaisuuksia vaikuttaa tutkimuksellani. Olen päässyt jakamaan tutkimukseni hedelmiä esimerkiksi valtakunnallisen Innokas-verkoston järjestämissä opettajien koulutuksissa, ohjelmoinnin opetuksen kehittämistyössä, kuten Uudet lukutaidot -kehittämisohjelmassa (www.uudetlukutaidot.fi), ja opettajien peruskoulutuksessa yliopistolla.

Uusien pedagogisten ideoiden jalkauttaminen ei ole kuitenkaan aina helppoa. On ollut ikävää havaita esimerkiksi, että ohjelmointi jäi vain valinnaiseksi aineeksi lukion uudessa opetussuunnitelmassa (Opetushallitus 2019). Vaikka ohjelmointi kuuluu peruskoulun opetussuunnitelmassa (Opetushallitus 2014) kaikille, koulut ovat yleisesti ottaen valitettavan ahtaalla. Tästä kielivät varmasti monet seikat niin opettajien osaamisessa, koulun johtamisessa kuin koulujen saamassa tuessakin.

Haluankin rohkaista kaikkia kunnioitettuja kasvatusalan toimijoitamme astumaan rohkeasti kohti tätä ehkä vieraankin oloista aihepiiriä. Aihepiiri ei pelota lapsia ja nuoria, mistä väitöskirjani kansikuvassa näkyvä, kanssani ohjelmoiva 3-vuotias tyttäreni on erinomainen esimerkki. Opettajia rohkaisen tekemään edes jotain ohjelmointiin liittyvää omien oppilasryhmiensä kanssa. Muistutan, että ohjelmoinnin tarkoituksena ei ole ainoastaan opettaa ohjelmoimaan, vaan myös ymmärtämään ja kriittisesti tarkastelemaan todellisen maailman ohjelmoitujen asioiden hyötyjä ja huolia (Mertala ym. 2020). Tällekin ajankohtaiselle aiheelle olisi tärkeä löytää jalansija, josta käsin se läpivalaisisi yleissivistävää koulutusta riittävän laajasti: jalansija, josta käsin ohjelmointi nähdään sekä tärkeänä oppimisen kohteena (Denning & Tedre 2019) että mainiona työskentelytapana, joka voi sytyttää luovan ongelmanratkaisun kipinän (Brennan & Resnick 2012). Nykyisessä työssäni kansainvälisessä ICILS-tutkimuksessa pääsenkin tarkastelemaan, kuinka valmiita tämän päivän oppilaat ovat opiskelemaan, työskentelemään ja elämään digitaalisessa maailmassa.

Väitöskirja

Fagerlund, Janne 2021. *Teaching, learning and assessing computational thinking through programming with Scratch in primary schools*. Jyväskylä: University of Jyväskylä. http://urn.fi/URN:ISBN:978-951-39-8882-1

Kirjallisuus

Barr, Valerie & Stephenson, Chris 2011. Bringing computational thinking to K–12: What is involved and what is the role of the computer science education community? *ACM Inroads* 2 (1), 48–54. https://doi.org/10.1145/1929887.1929905

Ben-Ari, Mordechai 1998. Constructivism in computer science education. *ACM SIGCSE Bulletin* 30 (1), 257–261. https://doi.org/10.1145/274790.274308

Black, Paul & Wiliam, Dylan 2009. Developing the theory of formative assessment. *Educational Assessment, Evaluation and Accountability* 21 (1), 5–31. https://doi.org/10.1007/s11092-008-9068-5

Brennan, Karen & Resnick, Mitchel 2012. *New frameworks for studying and assessing the development of computational thinking*. Paper presented at the meeting of AERA 2012, Vancouver, BC.

Denning, Peter & Tedre, Matti 2019. *Computational thinking*. Cambridge, MA: MIT Press Ltd. https://doi.org/10.7551/mitpress/11740.001.0001

Denning, Peter 2017. Remaining trouble spots with computational thinking. *Communications of the ACM* 60 (6), 33–39. https://doi.org/10.1145/2998438

Fagerlund, Janne, Häkkinen, Päivi, Vesisenaho, Mikko & Viiri, Jouni 2021a. Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education* 29 (1), 12–28. https://doi.org/10.1002/cae.22255

Fagerlund, Janne, Häkkinen, Päivi, Vesisenaho, Mikko & Viiri, Jouni 2021b. Assessing 4th grade students' computational thinking through Scratch programming projects. *Informatics in Education* 19 (4), 611–640. https://doi.org/10.15388/infedu.2020.27

Fagerlund, Janne, Vesisenaho, Mikko & Häkkinen, Päivi (arvioinnissa). Fourth grade students' computational thinking in pair programming with Scratch: A holistic case analysis.

Falkner, Katrina, Sentance, Sue, Vivian, Rebecca, Barksdale, Sarah, Busuttil, Leonard, Cole, Elizabeth, Liebe, Christine, Maiorana, Francesco, McGill, Monica & Quille, Keith 2019. An international comparison of K–12 computer science education intended and enacted curricula. Teoksessa Ihantola Petri & Falkner Nick (toim.), *Proceedings of the 19th Koli calling international conference on computing education research* (Koli calling '19). New York, NY: ACM. https://doi.org/10.1145/3364510.3364517

Fraillon, Julian, Ainley, John, Schulz, Wolfram, Friedman, Tim & Duckworth, Daniel 2020. *Preparing for life in a digital world. IEA international computer and information literacy study 2018 international report*. IEA. Switzerland, Cham: Springer. https://doi.org/10.1007/978-3-030-19389-8

Grover, Shuchi, Bienkowski, Marie, Basu, Satabdi, Eagle, Michael, Diana, Nicholas & Stamper, John 2017. A framework for hypothesis-driven approaches to support data-driven learning analytics in measuring computational thinking in block-based programming. Teoksessa Wise, Alyssa, Winne, Phillip & Lynch, Grace (toim.), *Proceedings of the seventh international learning analytics & knowledge conference* (LAK '17). New York, NY: ACM, 530–531. https://doi.org/10.1145/3027385.3029440

Grover, Shuchi & Pea, Roy 2013. Computational thinking in K–12: A review of the state of the field. *Educational Researcher* 41 (1), 38–43. https://doi.org/10.3102/0013189X12463051

Heintz, Fredrik, Mannila, Linda & Färnqvist, Tommy 2016. A review of models for introducing computational thinking, computer science and computing in K–12 education. Teoksessa *2016 IEEE frontiers in education conference* (FIE). IEEE, 1–9. https://doi.org/10.1109/FIE.2016.7757410

Hsu, Ting-Chia, Chang, Sshao-Chen & Hung, Yu-Ting 2018. How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education* 126, 296–310. https://doi.org/10.1016/j.compedu.2018.07.004

Kaarakainen, Meri-Tuulia, Kaarakainen, Suvi-Sadetta, Tanhua-Piiroinen, Erika, Viteli, Jarmo, Syvänen, Antti & Kivinen, Antero 2017. *Digiajan peruskoulu 2017 – Tilannearvio ja toimenpidesuositukset*. Valtioneuvoston selvitys- ja tutkimustoiminnan julkaisusarja 72/2017. Helsinki: Valtioneuvoston selvitys- ja tutkimustoiminta.

Leino, Kaisa, Rikala, Jenni, Puhakka, Eija, Niilo-Rämä, Mikko, Sirén, Marjo & Fagerlund, Janne 2019. *Digiloikasta digitaitoihin. Kansainvälinen monilukutaidon ja ohjelmoinnillisen ajattelun tutkimus* (ICILS 2018). Jyväskylä: Jyväskylän yliopiston Koulutuksen tutkimuslaitos.

Lonka, Kirsti, Kruskopf, Milla & Hietajärvi, Lauri 2018. Competence 5: Information and communication technology (ICT). Teoksessa Lonka, Kirsti (toim.), *Phenomenal Learning from Finland*. Helsinki: Edita, 129–150.

Maloney, John, Resnick, Mitchel, Rusk, Natalie, Silverman, Brian & Eastmond, Evelyn 2010. The Scratch programming language and environment. *ACM Transactions on Computing Education* 10 (4), 1–15. https://doi.org/10.1145/1868358.1868363

Mertala, Pekka, Palsa, Lauri & Slotte Dufva, Tomi 2020. Monilukutaito koodin purkajana: Ehdotus laaja-alaiseksi ohjelmoinnin pedagogiikaksi. *Media & Viestintä* 43 (1), 21–46. https://doi.org/10.23983/mv.91079

Opetushallitus 2014. *Perusopetuksen opetussuunnitelman perusteet 2014*. Helsinki: Opetushallitus.

Opetushallitus 2019. *Lukion opetussuunnitelman perusteet 2019*. Helsinki: Opetushallitus.

Wing, Jeannette Marie 2006. Computational thinking. *Communications of the ACM* 49 (3), 33–35. https://doi.org/10.1145/1118178.1118215

KT Janne Fagerlund *työskentelee Jyväskylän yliopiston Koulutuksen tutkimuslaitoksella projektitutkijana.*

 https://doi.org/10.33350/ka.111888