OHJELMOINNIN OPETUS JA SÄHKÖINEN TENTTI YLIOPISTON ENSIMMÄISILLÄ OHJELMOINTIKURSSEILLA

Turun yliopisto
University of Turku

Pro Gradu - tutkielma
Turun yliopisto
Tulevaisuuden teknologioiden laitos
Tietojenkäsittelytiede
Toukokuu 2018
Aleksi Mäkinen
505973

UNIVERSITY OF TURKU

Faculty of Science and Engineering / Department of Future Technologies

MÄKINEN, ALEKSI: Programming teaching and electronic examinations during university's first coding classes

Master's Thesis, 60 p., Computer science

May 2018

Teaching students to program is deemed to be very difficult. First programming classes suffer from big dropout rates and poor results worldwide. In an industry mostly populated by men women are a minority for reasons unknown. Also skill differences are big during the first programming classes.

There are many ways to combat the big dropout rates on the first programming classes. Different teaching methodologies like flipped classroom, couple programming and active learning are proposed as a solution. There have been good results of these altered teaching methods. Although the driving force behind the results has been credited to the change itself on the course and to the teachers willingness to improve their teaching. Learning systems are also used during some programming classes which will atleast partially help in the teaching for example through automated assessment. In doing so it will alleviate some pressure and workload from the teachers.

In this didactic IT thesis teaching programming is firstly examined on the common level and then moved on to things which might influence learning to program like motivation and cognitive skills. Ways to evaluate learning outcomes are examined through written, spoken and groupwork examinations. Whatever the evaluation method may be one can always use an electronic examination in the final assessment. Foremost in the programming environment where the computer is a natural part of the process.

In the research part about the Turku university's first programming course it was found out that students prefer computer tests over written ones on pen and paper and that activity during the course has at least a minor effect on the outcome of the summative examination. In general students were pleased with the learning environment used and the students felt that the course was succesfull. Using the learning system throughout the whole course all the way to final examination has alleviated some pressure from the students from learning to use a new system. Only problems that occured were due to big load to the system during the final examination.

Keywords: electronic exam, learning system, programming, teaching programming

TURUN YLIOPISTO

Luonnontieteiden ja Tekniikan tiedekunta /Tulevaisuuden teknologioiden laitos

MÄKINEN, ALEKSI: Ohjelmoinnin opetus ja sähköinen tenttiminen yliopiston ensimmäisillä ohjelmointikursseilla

Pro gradu –tutkielma, 60 s., Tietojenkäsittelytiede

Toukokuu 2018

Ohjelmoinnin opetuksen yliopistotasolla on todettu olevan vaikeaa. Ensimmäiset ohjelmointikurssit kärsivät maailmanlaajuisesti suurista keskeytysprosenteista ja huonoista tuloksista. Miesvaltaisella alalla naiset ovat vähemmistössä, syystä josta ei ole täyttä varmuutta ja lisäksi tasoerot ensimmäisillä ohjelmointikursseilla ovat suuria.

Lääkkeeksi ensimmäisten ohjelmointikurssien suuriin keskeytysprosentteihin on kehitetty erilaisia vaihtoehtoja. Erilaisia opetustyylejä kuten flipped classroom, pariohjelmointi ja aktiivinen oppiminen on ehdotettu ratkaisuiksi. Muutetuista opetustyyleistä on myös saatu hyviä tuloksia. Ajavana voimana on kuitenkin todettu olevan muutos itse kurssissa ja opettajan halu parantaa ohjelmoinnin opetusta. Oppimisjärjestelmät ovat myös käytössä osassa ohjelmoinnin kursseista. Nämä oppimisjärjestelmät osaltaan auttavat ohjelmoinnin opetuksessa mm. automaattisen arvioinnin myötä ja keventävät näin opettajien työtaakkaa.

Tässä didaktisen tietotekniikan tutkielmassa perehdytään ensin yleisellä tasolla tapoihin opettaa ohjelmointia sekä asioihin, jotka vaikuttavat ohjelmoinnin opiskeluun, kuten motivaatio ja kognitiiviset taidot. Tarkastellaan myös tapoja arvioida oppimistuloksia joko kirjoitetussa, suullisessa tai ryhmätyön muodossa. Arviointitavan ollessa mikä vain, voidaan käyttää sähköistä tenttiä lopullisessa arvioinnissa. Etenkin ohjelmoinnissa sähköinen tentti on olennainen, sillä ohjelmointi tapahtuu luontaisesti tietokoneella.

Tutkittaessa Turun yliopiston ensimmäistä ohjelmointikurssia saatiin tulokseksi, että opiskelijat suorittavat ohjelmoinnin tentin erittäin mielellään tietokoneella, ja että kurssiaktiivisuudella on ainakin lievä vaikutus tenttituloksiin. Yleisesti käytettyyn oppimisjärjestelmään oltiin tyytyväisiä ja opiskelijat kokivat kurssin onnistuneeksi. Lisäksi kokonaisvaltainen oppimisjärjestelmän käyttö läpi kurssin, tenttiä myöten, on osaltaan poistanut mahdollista ahdistusta uuden järjestelmän opiskelusta. Ainoat ongelmat aiheutuivat järjestelmän suuresta kuormituksesta tentin aikana.

Asiasanat: sähköinen tentti, oppimisjärjestelmä, ohjelmointi, ohjelmoinnin opetus

Sisällys

1 Johdanto

Viimeisten vuosien aikana ohjelmoinnin opetusta on tuotu yhä enemmän koko kansalle. Alemmilla asteilla on ryhdytty opiskelemaan ohjelmointia ja algoritmista ajattelua (Partanen ym. 2016). Reaktor niminen yritys on hiljattain tuonut koko kansalle avoimen tekoälykurssin, jolla Suomesta pyritään luomaan tekoälyn kärkimaa. Kaikella tällä pyritään sivistämään ihmisiä ohjelmoinnin ja ylipäätään IT:n mahdollisuuksiin. Yliopistotasolla on pitkään painittu ongelmallisten ohjelmointikurssien kanssa (Lahtinen ym. 2015). Iso osa ohjelmoinnin ensimmäisistä kursseista kärsii joko huonoista lopputuloksista tai suurista keskeytysprosenteista maailmanlaajuisesti (Watson&Li 2014, Bennedsen&Caspersen 2007). Opiskelijat eivät tahdo ymmärtää abstrakteja malleja, eikä heillä välttämättä ole aiempaa kokemusta ohjelmoinnista (Vihavainen ym. 2014). Näin tasoerot ovat suuret ja tämä osaltaan asettaa opettamiselle haasteita. Miesvaltainen ala kärsii myös osittain naisten vähäisestä määrästä (Lishinski ym. 2016).

Ohjelmoinnin opetus on kaivannut uusia tuulia vanhojen juurtuneiden tapojen tilalle. Turun yliopistossa muutos alkoi ViLLE oppimisjärjestelmän käyttöönoton myötä vuonna 2011. Ensimmäiselle ohjelmointikurssille implementoitiin tämän jälkeen mm. aktiivista oppimista, pariohjelmointia sekä sähköinen tentti.

Tässä tutkielmassa tullaan perehtymään siihen, mitkä asiat vaikuttavat ohjelmoinnin opiskeluun. Mukaan luetaan oppimisympäristö kuin taustavaikuttajatkin, kuten motivaatio ja kognitiiviset taidot. Tämän lisäksi perehdytään mahdollisiin, hieman perinteisistä poikkeaviin opetustyyleihin, kuten flipped classroom. Käsitellään myös muita ohjelmoinnin opetuksessa oleellisia asioita, jotka ovat opetusteknologiat sekä sähköinen tenttiminen. Sähköinen tenttiminen liittyy kiinteästi ohjelmointiin, sillä mikään muu yliopistossa opiskeltava aihe ei tapahdu näin kokonaisvaltaisesti tietokoneella kuin ohjelmointi. Lopussa tutkimus, jossa tutkitaan kuinka Turun yliopiston ensimmäisellä ohjelmointikurssilla kurssiaktiivisuus vaikuttaa kurssin lopputulokseen, ja kuinka opiskelijat suhtautuvat sähköiseen tenttimiseen. Tutkimuskysymyksiä joita tullaan käsittelemään, on kolme kappaletta:

1) Vaikuttaako kurssiaktiivisuus tentin lopputulokseen?
2) Minkä tasoinen Algoritmien ja ohjelmoinnin peruskurssin päättökoe on?
3) Millainen suhtautuminen opiskelijoilla on ohjelmoinnin sähköiseen tenttimiseen?

Tutkielma rakentuu ohjelmoinnin opetuksesta, jonka sisällä käsitellään oppimisympäristöä, ohjelmoinnin opiskelun taustavaikuttajia, opetustyylejä, opetusteknologioita sekä opetusmetodeja. Tästä siirrytään opetuksen arviointiin ja sähköiseen tenttimiseen. Lopuksi

suoritetaan tutkimus, jossa käsitellään Turun yliopiston ensimmäisen ohjelmointikurssin kurssiaktiivisuutta, tenttituloksia ja tenttipalautetta.

2 Ohjelmoinnin opetus

Ensimmäisen yliopiston ohjelmointikurssin (tästä eteenpäin CS1) läpäisee maailmanlaajuisesti vain noin kaksi kolmesta opiskelijasta (Watson&Li 2014, Bennedsen&Caspersen 2007). Tämä tarkoittaa sitä, että noin joka kolmas opiskelija epäonnistuu kurssilla. Tarkkaa syytä tähän ei tiedetä mutta on spekuloitu, että internetin aikakaudella monet opiskelijat ovat menettäneet kiinnostuksensa perinteisiin luentoihin. Osittain tästä johtuen oppimistulokset ovat laskeneet ja opiskelumotivaatio vähentynyt. Ongelmaa vastaan on kehitetty monia erilaisia opetustyylejä, jotta oppimistulokset saataisiin paremmiksi. Näistä erityisesti opiskelijaan keskittyvät opetustyylit saivat Debiecin (2018) tutkimuksessa aikaan huomattavia parannuksia oppilaiden luento-osallistumisissa sekä oppimistuloksissa. Debiec esittää tutkimuksessaan, että opiskelijaan keskittyvillä oppimistyyleillä on tiettyjä esteitä:

1) E-materiaalien vaikea tuottaminen/saatavuus.

2) Merkittävä ajallinen panostus aktiiviseen internetkommunikaatioon opiskelijan kanssa.

3) Opiskelijoiden haluttomuus ottaa aktiivista roolia omasta koulutuksestaan.

4) Luennoitsijoiden yleiset luulot siitä, että opiskelijakeskeiset aktiviteetit eivät mahdollista keskustelua kaikista tarvittavista aiheista.

Debiec kuitenkin toteaa tutkimuksensa päätteeksi, että nämä ongelmat on mahdollista selättää yhdistelemällä eri opetustyylejä ja -metodeja kuten tutoriaaleja sekä lyhentämällä kurssin kestoa.

Vihavaisen ym. (2014) tutkimuksessa erilaisista opetustyyleistä löydettiin, että lähes mikä tahansa interventio opetuksessa kasvattaa kurssin läpäisyprosenttia kolmanneksella, verrattuna perinteisiin luentoihin ja laboratorioharjoituksiin. Interventioiden välille ei kuitenkaan löydetty tilastollisesti merkittäviä eroja. Tämän perusteella arveltiin, että muutos on ajava voima opetusinterventioiden takana. Muutoksen takia opiskelijat saavuttavat parempia oppimistuloksia. Kyseiset interventiot tullaan käsittelemään tarkemmin luvussa 2.5. Tutkimuksissa ei eritelty erikseen naisia ja miehiä mutta Rubion ym. (2015) tutkimuksessa käsiteltiin kyseistä aihetta.

Ohjelmointi on miesvaltainen ala jossa naiset ovat vähemmistössä, syystä josta ei ole täydellistä varmuutta. Rubio ym. (2015) tutkivat naisten ja miesten eroja ohjelmoinnin oppimisessa yliopiston CS1 kursseilla. Heidän tutkimuksensa mukaan miesten ja naisten erot ohjelmoinnin oppimisessa johtuvat eri näkemyksistä sekä erilaisista oppimisen lopputuloksista. Miehet kokivat ohjelmoinnin helpommaksi, he aikovat todennäköisemmin ohjelmoida tulevaisuudessa kuin naiset. Miesten oppimistulokset ovat myös parempia kuin naisilla. Tämän väitettiin johtuneen

opetustyyleistä ja tutkimuksessa todettiin, että heidän kehittämänsä oppimismoduulit, jotka noudattivat aineellisen ohjelmoinnin periaatteita, vähensivät ja jopa häivyttivät sukupuolien välisen eron ohjelmoinnissa.

Kuinka voidaan määritellä, kuka hallitsee ohjelmoinnin ja kuka ei? Miten luokitellaan mitä tulisi oppia? Winslown (1996) mukaan ohjelmoinnin opiskelijat ja taitajat voidaan jakaa viiteen eri luokkaan. Nämä luokat ovat

1) Noviisi: Oppii objektiivisia faktoja ja ominaisuuksia sekä sääntöjä, joilla pystyy päättämään toimintoja perustuen näihin faktoihin ja ominaisuuksiin (konteksti vapaata).
2) Kehittynyt aloittelija: Alkaa tunnistamaan ja käsittelemään tilanteita, joita annetut faktat, ominaisuudet ja säännöt (konteksti riippuvaiset) eivät käsittele. Kuitenkaan täysin ymmärtämättä mitä tekee.
3) Kyvykäs: Pohdittuaan tilannetta tietoisesti, valitsee järjestelmällisen suunnitelman tavoitteen saavuttamiseksi.
4) Taituri: Ei tarvitse enää tietoisesti käydä läpi kaikkia askelia laatiakseen suunnitelmaa.
5) Asiantuntija: Asiantuntija yleensä tietä kuinka toimia perustuen kehittyneeseen ja harjoitettuun ymmärtämiseen.

Winslown mukaan yliopiston ensimmäinen tutkinto riittää noin kyvykkään ja taiturin välimaastoon. Asiantuntijaksi tullakseen vaatisi vähintään 10 vuoden kokemuksen alasta, joten tämä ei täten olisi saavutettavissa perinteisesti yliopistoissa.

Robins ym. (2003) puolestaan esittää, että ohjelmoinnin opiskelijat voidaan noviisitasolla jakaa kahteen eri ryhmään: tehokkaat ja tehottomat noviisit, joiden erona voidaan pitää tehokkaiden kykyä oppia ohjelmoimaan ilman suurempaa avustusta toisin kuin tehottomat noviisit. Robins ym. esittää että ohjelmoinnin opetuksen aloituksessa tulisi keskittyä enemmän saamaan noviiseista tehokkaita oppijoita. Suurimpana erona näiden kahden välillä ei Robinsin mukaan pidetä tietoa vaan oppimisen strategioita, joihin perehtymällä voitaisiin tehottomista noviiseista mahdollisesti tehdä tehokkaita noviiseja ja näin helpottaa ohjelmoinnin oppimista. Yksi osatekijä, jolla oppimiseen voidaan vaikuttaa, on oppimisympäristö.

2.1 Oppimisympäristö (online vs. paikan päällä)

Oppimisympäristönä voidaan tarkoittaa paikalla tapahtuvaa oppimista ns. fyysistä opiskelua tai opiskelua internetin välityksellä. Kailan ym. (2015) tutkimuksen mukaan ohjelmoinnin opetuksen muututtua Turun yliopistossa perinteisestä luento-opetuksesta enemmän aktiivisen oppimisen suuntaan vuosien 2011-2012 välillä useampi opiskelija läpäisi kurssin. Käyttöönotettiin ViLLE -oppimisjärjestelmä, ja osa luennoista muutettiin tutoriaaleiksi. Tämän myötä arvosanojen keskiarvo pysyi samana mutta useampi opiskelija läpäisi kurssin. Kyseisessä opetustavassa yhdistetään paikalla tapahtuva oppiminen internetin välityksellä tapahtuvaan oppimiseen, sillä opiskelija saattoi tehdä tehtäviä oppimisjärjestelmässä myös luentojen tai tutoriaalien ulkopuolella. Joissain tapauksissa pelkästään internetin välityksellä tapahtuva oppiminen on mahdollista esimerkiksi pitkien välimatkojen vuoksi.

Yerby&Floyd (2013) tutkivat kuinka täysin internetin välityksellä järjestettävä IT-koulutus eroaa vastaavasta perinteisestä toteutustavasta. Tutkimuksen lopputulokseksi saatiin, että oikein toteutettuna eroa ei ole kummankaan hyväksi ja internetin välityksellä tapahtuva opetus oli yhtä tehokasta kuin paikan päällä tapahtunut opiskelu. Internet kurssin järjestäminen vaatii enemmän kuin pelkän tekstikirjan valitsemista ja kurssin rakentamista tämän päälle. On tärkeää, että suunnittelu on yksityiskohtaista, sisältää tarkkojen tavoitteiden suunnittelun koko kurssille, kuten jokaiselle yksittäiselle luennolle sekä tarkennetaan tehtävienantoa yksityiskohtaiseksi ja kuvaillaan tarkat lopputulokset (Dykman&Davis 2008).

Ainoastaan internetin välityksellä tapahtuvaa oppimista ovat tutkineet myös mm. Ferdianto&Desak (2017) jotka painottavat, että mikäli opiskelu tapahtuu ainoastaan internetin välityksellä tulisi tämän tapahtua oppimisjärjestelmässä. Heidän tutkimuksessaan vertailtiin erilaisia opetustyylejä internetin välityksellä tapahtuvassa oppimisessa ja päädyttiin kolmeen neuvoon, joita opettajien tulisi noudattaa. Ensimmäisenä opettajien tulee olla aktiivisia ja responsiivisia, esimerkiksi keskustelufoorumeilla. Toisena opettajan tulisi kiinnittää huomiota ryhmätyöhön ja keskusteluun, jotta opiskelijat pystyvät kehittämään ns. pehmeitä taitoja. Kolmantena opettajan tulisi olla kurinalainen korjatessaan tehtäviä, jotta opiskelijat saisivat hyvää palautetta ja pystyisivät näin oppimaan virheistään.

MOOC kurssit ovat oma asiansa opiskelun saralla. Kyseisillä kursseilla voi olla kymmeniä tuhansia opiskelijoita, joka puolestaan asettaa järjestelmälle tiettyjä vaatimuksia. Aarabi ym. (2016) tutkivat kuinka, yli 30 000 opiskelijan sovelluskehityskurssilla opetus tulisi järjestää. He löysivät seitsemän eri ehdotusta MOOC kurssin järjestämiselle.

1) Vältetään ylimääräistä tarinankerrontaa ja keskitytään tarkasti käsiteltävään asiaan.
2) Vastataan avainkysymyksiin sen sijaan että opetettaisiin ongelman ratkontaa.
3) Luodaan materiaalia kaiken ikäisille opiskelijoille ja otetaan kaiken ikäiset huomioon materiaalia luodessa.
4) Keskimääräinen opiskelijan keskittymisaika on noin 1-2minuuttia, jaetaan opetusmoduulit tämän mukaan.
5) Tehtävät antavat opiskelijoille kuvan siitä hallitsevatko he opetetun asian. Keskitytään että tehtävät ovat oikean tasoisia, eivät liian helppoja eivätkä liian vaikeita.
6) Huomioidaan että luentoja katsotaan usein pieniltä kännykän ruuduilta, rajoitetulla kaistalla ja meluisassa ympäristössä.
7) Tehokkuus on tärkeää, sillä opiskelijat tulivat oppimaan tietyn rajatun asian, jonka he voisivat helposti etsiä Googlesta. Eli kilpailijana voidaan tässä suhteessa pitää Googlea.

Onnistuneen online kurssin luomista voidaan siis pitää haasteellisena sen vaatiman uuden opettelun myötä. Se mitä voidaan pitää onnistuneena kurssina, voidaan tarkastella mm. Grahamin ym. (2001) esittämän seitsemän käytännön linssin läpi. Kyseisiin arviointimenetelmiin voi perehtyä tarkemmin Grahamin ym. (2001) tutkimuksessa. Tutkimuksessa mainitaan mm. aikarajojen ja selkeiden tavoitteiden asettaminen kurssille.

2.2. Ohjelmoinnin opiskelun taustavaikuttajat

Seuraavaksi tullaan käymään läpi ohjelmoinnin opiskeluun liittyviä taustatekijöitä, jotka ovat ikä ja sukupuoli, motivaatio sekä kognitiiviset taidot. Kyseiset asiat eivät ole yksinään ohjelmoinnin taustavaikuttajia, vaan voivat koskea myös muuta opiskelua.

2.2.1 Sukupuoli ja ikä

Bergin&Reillyn (2005) tutkimuksen mukaan suurimmat osatekijät jotka vaikuttivat ohjelmoinnin kurssin lopputulokseen, olivat kurssin ns. mukavuusaste, opiskelijan käsitys heidän ymmärryksestään, aikaisemmat suoritukset matematiikasta ja sukupuoli. Miehiä valmistuu ohjelmoinnin tutkinnoista neljä kertaa enemmän kuin naisia (Lishinski ym. 2016). Syyt naispuolisten opiskelijoiden vähyyteen voidaan liittää roolimallien puutteeseen, erilaisuuksiin aikaisemmissa ohjelmointi kokemuksissa, vihamieliseen kulttuuriin ja erilaisiin uupumisasteisiin.

Jotta naiset saataisiin kiinnostumaan ohjelmoinnista, on vanhoja opetusmenetelmiä kehitetty eteenpäin. Esimerkkinä Rubion ym. (2015) tutkimus, jossa käytettiin ensimmäisillä

ohjelmointikursseilla kontekstualisoitua ohjelmointia, Arduino-alustaa hyväksikäyttäen. Perinteisin menetelmin suoritetussa kurssissa naisopiskelijoiden käsitys ohjelmoinnista oli alhaisempi kuin miehillä. Ohjelmointikurssien tuloksissa oli myös eroja miesten hyväksi, vaikka ne eivät olleetkaan tilastollisesti merkittäviä. Muutetussa kurssissa, jossa käytettiin Arduino-alustaa apuna ohjelmoinnissa ja konkretisoitiin ohjelmointi, saatiin tulokset miesten ja naisten välillä hävitettyä. Tämä tuki olettamusta siitä, että fyysinen ja visualisoitu ohjelmointi auttavat naisia ohjelmoinnissa ja antavat heille itseluottamusta ohjelmoida. Rubio ym. (2015) toteavat, että muut tutkimukset, jotka kehittivät intervention naisopiskelijoiden ohjelmoinnin asenteiden parantamiseen (kts. Sabitzer&Pasterk 2014 ja J.Freeman ym. 2014), olivat onnistuneita.

Ikä on toinen muuttuja, jota on tutkittu mm. Kailan ym. (2017) toimesta yliopisto-opiskelijoiden ja yläasteopiskelijoiden välillä. Tulokset osoittivat, että iällä ei ollut suurta merkitystä lopputuloksissa, mutta opiskelumetodit ja ajan käyttö olivat hyvin erilaiset näiden kahden ryhmän välillä. Yläaste ikäiset opiskelijat suorittivat kaikki tehtävänsä kouluajalla ja yliopisto-opiskelijat suorittivat tehtäviä pitkin päivää. Myös palautusten määrässä oli eroja.

Ala-aste ikäisille suoritettuja tutkimuksia ei löytynyt tätä tutkimusta varten, mutta Hijón-Neira ym. (2017) tutkivat Madridin ala-asteiden ohjelmoinnin opetusta kouluille lähetettävien kyselyiden avulla. Saatujen vastausten perusteella 39.1% kouluista ei opeta ohjelmointia joko aikataulujen takia tai puuttuvien opettajien vuoksi. Ala-asteissa joissa lapsia opetettiin, käytettiin pääosin pelejä tai Scratchia ilman tarkkaan mietittyä suunnitelmaa tai metodologiaa. Tarjottaessa kouluista 61% olisi valmis toteuttamaan tarjottua metodologiaa opetuksessaan.

2.2.2 Motivaatio

Oppiminen ja motivaatio ovat monimutkaisia asioita käsitellä ja tutkia. Opiskelijoiden motivaatio syntyy eri lähteistä, toisilla ohjelmoinnin opiskelu voi olla väylä parempaan tai ”hyvään” työpaikkaan, toisilla opiskelu tapahtuu omasta ilosta ja halusta. Usein on puhuttu ulkoisesta ja sisäisestä motivaatiosta, jotka ajavat opiskelijoita eteenpäin. Seuraavaksi tullaan perehtymään näihin ja muihin motivaation lähteisiin ohjelmoinnin opiskelussa.

Motivaatiota voitaneen kuvata yrityksen määränä, jonka avulla saavuttaa tietyn maalin ja oppimismotivaationa voidaan pitää jatkuvan työn määrää, jonka opiskelija laittaa opiskeluunsa (Law ym. 2010). Moni asia voi vaikuttaa motivaation syntyyn. Motivaation lähde voi olla sisäinen tai ulkoinen. Henkilön tunteet ja elämäntilanne voivat vaikuttaa motivaatioon ja opiskelun määrään. Sosiaaliset suhteet ja halu miellyttää voivat olla myös osatekijöinä motivaation synnyssä.

Jenkins (2001) kuvaa tutkimuksessaan epävirallisen listan mitattavista motivaation lähteistä:

- opiskelijan halu miellyttää opettajaa
- opiskelijan huomattava tarve opetusmateriaalille
- mielenkiinto opiskeltua aihetta kohtaan
- oppijan filosofiset arvot ja asenteet
- oppijan asenne opetusmateriaalia kohtaan
- tuleva työ- tai akateeminen ura
- sytykkeet ja palkinnot jotka nousevat opiskelun myötä

Jenkins (2001) listaa geneerisiksi motivaation lähteiksi sisäisen ja ulkoisen motivaation sekä saavutukset. Sisäisellä motivaatiolla tarkoitetaan henkilön sisäisiä haluja opiskeltavaa kohdetta kohtaan. Ulkoisella motivaatiolla tarkoitetaan henkilön ulkopuolelta tulevia motivaation lähteitä. Saavutuksella tarkoitetaan kilpailullista pärjäämistä, jossa halutaan suoriutua hyvin ja (toisinaan) paremmin kuin muut.

Opiskelijoiden opiskelutavat saattavat muuttua suuresti riippuen motivaation lähteestä. Esimerkiksi ulkoisesti motivoitunut opiskelija saattaa motivoitua ainoastaan tehtävistä saatavista pisteistä, jotka edistävät opiskeltua asiaa. Sisäisesti motivoitunut opiskelija taas saattaa motivoitua asioista, jotka eivät varsinaisesti liity asiaan, ja perehtyy aihepiiriin myös kurssin ulkopuolelta. Mikäli saavutus on pääasiallinen motivaation lähde, voi opiskelija käyttää opiskeltavaan aiheeseen niitä keinoja, jotka takaavat parhaat pisteet ja parhaan arvosanan kurssilta.

Ohjelmoinnin tutkinnon saavuttamiseksi opiskelijat motivoituvat Jenkinsin (2001) mukaan pääosin sisäiseksi oppimiseksi luokiteltavan oppimisen takia, ja ulkoisiksi luokiteltavien tavoitteiden takia. Oppimisella tarkoitetaan tässä tapauksessa sitä, että opiskelijat ovat motivoituneita saavuttamaan tutkinnon oman mielenkiintonsa vuoksi, eli he haluavat oppia. Tavoitteilla tarkoitetaan mahdollisen tulevan työpaikan saamista tai muuta vastaavaa. Kyseinen tutkimus suoritettiin yliopiston ensimmäisillä ohjelmointikursseilla aloitteleville opiskelijoille. Ensimmäisen ohjelmointikurssin läpäisemisen motivaationa oli noin puolella opiskelijoista kurssin pakollisuus, jota Jenkinsin mukaan voidaan pitää huolestuttavana, sillä ohjelmointi on tärkeä taito, mikäli aikoo työllistyä kyseiselle alalle. Syyt sille miksi opiskelijat olivat tulleet opiskelemaan kyseistä alaa, jakautuivat Jenkinsin tutkimuksen mukaan lähes tasan saavutuksen ja ulkoisen motivaation välille.

Lawn ym. (2010) mukaan ohjelmoinnin opiskelun sisäinen motivaatio syntyy pääosin henkilökohtaisista asenteista ja odotuksista, sekä haastavista tavoitteista. Ulkoista motivaatiota

edisti sosiaalinen paine ja kilpailu. Tutkimuksessa käytettiin sähköistä järjestelmää opiskelun apuna.

Vihavainen toteaa Settlen ym. (2014) paneelin alustuksessaan, että opiskelijan yrityksen määrä korreloi suoraan menestyksen kanssa, ja että motivaation puute on suuri syy uupumiseen ensimmäisillä ohjelmointikursseilla. Vihavaisen mukaan opettajan tulee ottaa huomioon antamansa ohjaus kursseilla. Esimerkiksi, mikäli opiskelija kamppailee ongelman kanssa opettajan antama vastaus ”Yritä parhaasi” -antaa huonommat tulokset, kuin jos opettaja antaisi selvän maalin siitä mitä opiskelijan tulisi tehdä. Esimerkkinä ”yritä saada tehtävä 80%-valmiiksi”.

Sorva kuvailee Settlen ym. (2014) paneelikeskustelun alustuksessaan kuinka ohjelmoinnin visualisointi vaikuttaa opiskelijan motivaatioon. Sorvan mukaan pelkkä visualisointi ei riitä, vaan opiskelijan tulee osallistua visualisointiin ja tehdä jotain merkittävää kyseisellä työkalulla. Esimerkkinä visualisoinnin manipulaatio ohjelmoimalla on tehokkaampaa, ja motivoi opiskelijoita enemmän kuin pelkän visualisaation katselu. Visuaaliset työkalut auttavat osaa opiskelijoita ymmärtämään, kuinka koodi toimii. Opettajan käyttäessä visualisaatiota työkaluna, tärkeää ei ole pelkästään visualisaation näkeminen, vaan se mitä opiskelija pystyy tekemään ohjelmointia apunaan käyttäen tälle visualisoinnille. On myös tärkeää, että opiskelija on tutustunut visualisointiin käytettävään työkaluun etukäteen, sillä tämä edistää oppimista ja oppiminen on tehokkaampaa verrattaessa siihen, kun opiskelija tutustuu kyseiseen työkaluun ensimmäistä kertaa (Laakso ym. 2008)

Motivaatio on yksi osa-alue ohjelmoinnin opiskelussa, vaikeasti mitattava ja ajasta riippuva määre jonka tutkiminen voi olla hyvinkin vaikeaa. On esitetty, että motivaatio kumpuaa sisäisestä tai ulkoisesta motivaatiosta ja lisäksi kolmantena on esitetty, että motivaatio syntyy saavutusten hakemisesta. Näiden kolmen välille ei voida vetää tarkkaa rajaa, mutta opettajan tulisi opetuksessaan ottaa huomioon erilaiset opiskelutyylit ja mahdolliset motivaation lähteet ja tätä kautta hyödyntää motivaatiota ohjelmoinnin opetuksessa. Mahdolliset työkalut saattavat auttaa opiskelussa ja parantaa motivaatiota oikein toteutettuna.

2.2.3 Kognitiiviset taidot

Kognitiivisilla taidoilla käsitellään tässä yhteydessä opiskeluun liittyviä taitoja, kuten opetetun asian hallinta ja asiakokonaisuuksien muodostaminen. Yksi osa kognitiivisia taitoja ovat Swellerin ym. (1994) mukaan skeemat. Skeemoilla tarkoitetaan mielen sisäisiä rakennelmia tosimaailman rakenteista. Tietoa järjestetään skeemoihin ja skeemoja käytetään uuden tiedon ymmärtämiseen. Käytännön esimerkkinä voidaan käyttää puuta, mikäli ihminen näkee puun

hänen ei tarvitse erikseen miettiä tai pohtia miltä puu näyttää, sillä hänelle on jo aikaisemmista kokemuksista muodostunut käsitys miltä puu näyttää. Näin henkilön ei tarvitse muistaa jokaista yksityiskohtaa kyseisestä puusta vain mieltääkseen sen puuksi. Muistellessa kyseistä puuta jo olemassa oleva skeema todennäköisesti vaikuttaa mielikuvaan miltä puu näyttää.

Opiskelijoilla skeemat helpottavat ymmärtämään asiakokonaisuuksia. Ymmärrettäessä yhden asian, tämän päälle rakentaminen suurella todennäköisyydellä helpottuu, sillä jo olemassa olevan mentaalisen mallin päälle on helpompi kasata lisätietoa. Eli skeeman omaksuminen helpottaa suuresti intellektuaalisen tiedon ymmärtämistä ja käsittelyä.

Tiukasti skeemoihin liittyy myös tiedon automaattinen prosessointi. Skeemoja omaksutaan pitkällä aikavälillä ja näistä skeemoista koostuva tiedon käyttöönotto riippuu tiedon automaattisesta prosessoinnista. Swellerin (1994) mukaan tietoa voidaan käsitellä automaattisesti tai kontrolloidusti. Esimerkkinä tämän gradun lukijat, jotka todennäköisesti pystyvät lukemaan kirjoitettua tekstiä ilman suurempaa ajatustyötä, mutta lukemista vasta harjoittelevat voivat joutua keskittymään jokaiseen sanaan ja kirjaimeen, sen sijaan että keskittyisivät siihen mitä sanat tarkoittavat. Sweller kuvaa tiedon automaattista käsittelyä toiseksi tärkeimmäksi oppimismekanismiksi heti skeemojen jälkeen. Tiedon automaattinen prosessointi kehittyy pikkuhiljaa tiedon karttuessa opiskellusta aihealueesta. Skeeman käyttöönotto riippuukin tiedon automaattisesta käsittelystä. Opiskelijalla voi olla jo skeema hallussa, mutta hän joutuu käyttämään tietoista työtä jonkin tietyn ongelman ratkaisuun. Ajan kuluessa tiedon käsittely automatisoituu ja henkilölle jää enemmän kognitiivista kapasiteettia muihin toimintoihin. Ilman automaatiota suoritukset ovat hitaita, kömpelöitä ja alttiita virheille. Tämän takia automaattinen tiedon prosessointi on tärkeä osa kognitiivisia taitoja (Sweller 1994).

Aikaisemmin tässä tutkielmassa on todettu, että ohjelmoinnin taitajat voidaan Winslown (1996) jakaa viiteen kategoriaan noviisista-asiantuntijaan. Winslow jakaa tutkimuksessaan Swellerin (1996) toteamuksen tiedon luonteesta siinä, että asiantuntija on rakentanut mieleensä mentaalisen mallin oikean maailman ongelman ratkaisusta ts. skeeman. Näin asiantuntijalla ei välttämättä ole enempää tietoa ohjelmointikielestä kuin aloittelijalla, mutta ongelmanratkaisukyvyt ja aihealueen ymmärrys auttavat asiantuntijaa ohjelmointi ongelman ratkaisemisessa.

Erityisesti ohjelmointiin liittyvät kognitiiviset taidot ovat Bergin&Reillyn (2005) mukaan ongelmanratkaisu, abstrakti ajattelu, ongelman välitystaidot, loogiset kyvyt ja ajattelutavat. Tutkittaessa ohjelmointia ja siihen liittyviä kognitiivisia taitoja, ei voida sulkea muita opiskeluun mahdollisesti liittyviä asioita, kuten minäpystyvyys ja henkilökohtaiset piirteet (Lishinski 2016).

2.3 Opetustyylit

Yliopistojen pääasiallisena opetustyylinä on jo noin 900 vuotta ollut perinteinen luento (Ruegg 2014). Suuret luokkakoot sekä suuret keskeyttämisprosentit ohjelmointikursseilla ovat ajaneet tutkijoita ja opettajia etsimään uudenlaisia opetustapoja korvaamaan perinteinen luento-opetus. Kyseisiä opetustyylejä on löydetty monia. Seuraavaksi tullaan käsittelemään opetustyylejä, joita voidaan käyttää mm. ohjelmoinnin opetuksessa.

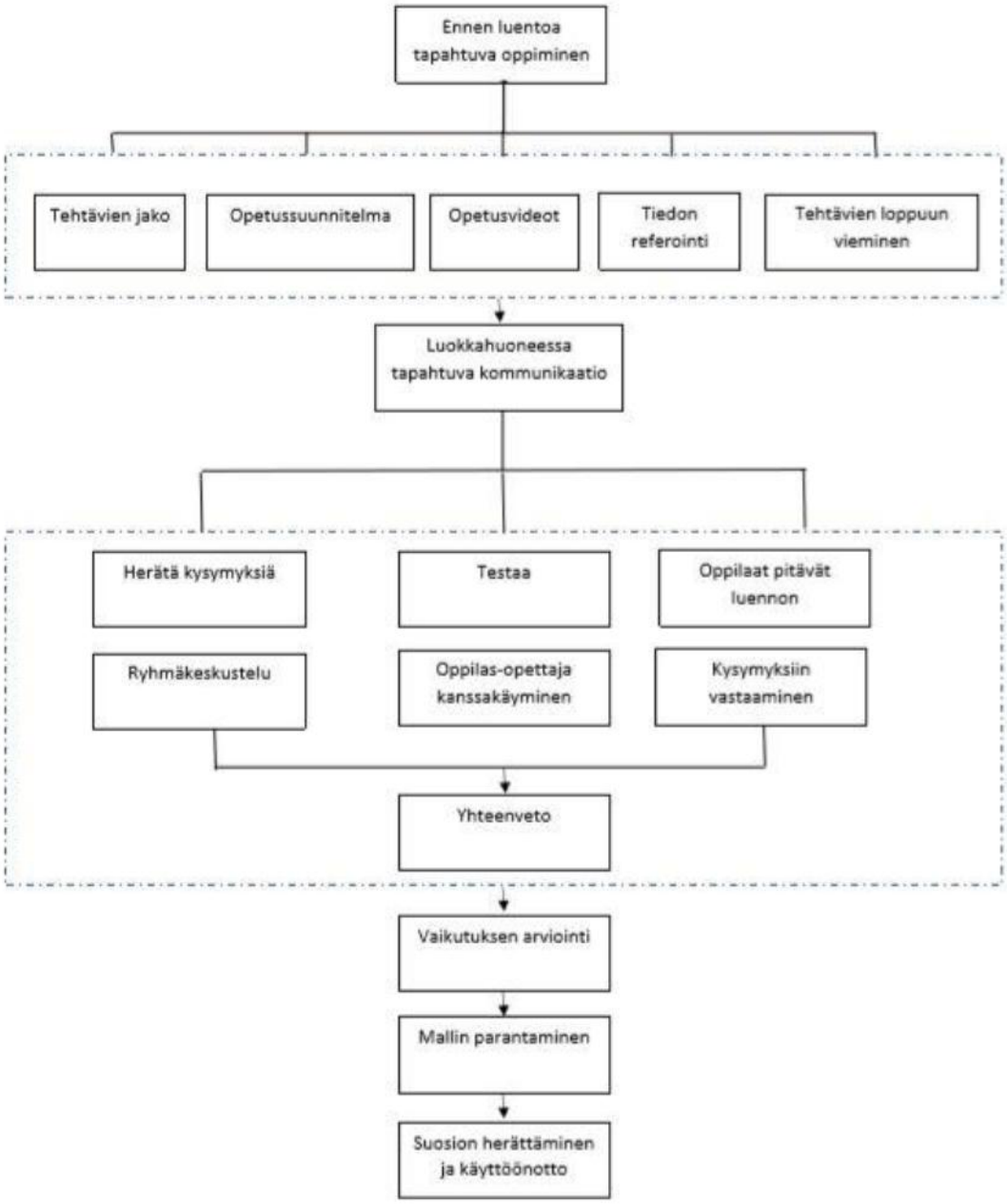
2.3.1 Flipped classroom

Käänteinen opetus, englanniksi flipped classroom, tässä mallissa opetus käännetään siten että opiskelija opiskelee luentojen aiheita ennen luentoa internetistä ja paikan päällä tapahtuvassa opetuksessa läpikäydään ennakkoon opiskeltu materiaali, ja mahdollisesti sovelletaan jo opittua tietoa. Ennakkoon opiskeltava materiaali on yleensä opettajan valmistama video tai muu materiaali. Luennoilla etukäteen opittua materiaalia sovelletaan mm. ongelmanratkontaan, keskusteluun tai palautteen antamiseen (kuva 1). Kyseinen malli siirtää oppimisen opettajakeskeisestä mallista opiskelijakeskeiseen malliin. Näin opiskelijoille itselleen syntyy suurempi vastuu opiskelusta. Tätä voidaan pitää huonona tai hyvänä asiana riippuen opiskelijasta ja hänen motivaatiotasostaan. Tämä kuitenkin mahdollistaa opiskelijoille oman tahdin opiskeluun. Opiskelijat jotka hallitsevat käsitellyn alueen voivat siirtyä omaan tahtiinsa vaikeampiin tehtäviin, ja he joilla on ongelmia voivat tehdä enemmän toistoja valmistautuessaan luentoihin, ja luennoilla kysyä neuvoja opettajalta. Samalla opettajalle vapautuu aikaa implementoida luennoille erilaisia opetusstrategioita riippuen jokaisen oppilaan tarpeesta (Jungiang ym. 2017).

Alhazbi (2016) tunnisti tutkiessaan käänteisen opetuksen vaikutusta ohjelmoinnin opetteluun, että suurin haaste tässä opetustavassa on, kuinka saada opiskelija aktiivisesti tutustumaan ennen luentoa annettuun materiaaliin ja näin tulemaan luennolle valmistautuneena. Elmaleh&Sankaramanin (2017) tutkimuksen mukaan opiskelijat käyttävät noin tunnin enemmän aikaa käänteisessä opetustavassa opetetussa ohjelmoinninkurssissa kuin perinteisessä tavassa. Tämän ei kuitenkaan todettu vaikuttavan negatiivisesti opiskelijoiden motivaatioon tai asenteisiin.

Useammassa tutkimuksessa todetaan, että käänteinen opetus parantaa koetuloksia ja motivoi opiskelijoita enemmän kuin perinteinen luento-opetus (Elmaleh&Sankaraman 2017, Jungiang

ym. 2017, Yan&Cheng 2017, Trpkovska 2017). Indi (2016) tutki kuinka käänteinen opetus vaikuttaa implementoitaessa kolmannen vuoden insinöörien Java- ohjelmointikurssiin. Hän löysi tutkimuksessaan että 92-98% opiskelijoista piti käänteisestä opetustyylistä ja haluaisi sen implementoituvan myös muihin kursseihin. Kurssin myötä opiskelijoiden itseoppimiskapasiteetti parani ja 100% opiskelijoista oli sitä mieltä, että kurssia varten luodut opetusvideot olivat hyödyllisiä. Käänteisen opetustyylin onnistuminen tuntuu riippuvan suuresti tuotetun materiaalin tasosta, ja siitä kuinka kaikilla opiskelijoilla on pääsy kyseiseen materiaaliin. Tämä luo opettajille uudenlaista painetta luoda materiaalia ja muokata opetustaan enemmän valmentavaan muotoon kasvokkain tapahtuvaa opetusta varten.



Kuva 1 flipped classroom

2.3.2 Pariohjelmointi

Pariohjelmoinnissa ohjelmointi hoidetaan pareittain. Toinen henkilöistä kirjoittaa koodia (”driver/kuski”) ja samalla toinen henkilö auttaa ja seuraa vierestä (”navigator/navigoija”). Yleensä kuski hallitsee tietokoneen hiirtä ja näppäimistöä ja navigoija tutkii, tarkkailee ja antaa ideoita kuskille. Tästä opetustavasta on saatu ristiriitaisia tuloksia. Rajala ym. (2011) tutkimuksen mukaan opiskelijat käyttivät pareittain enemmän aikaa ohjelmointitehtävien tekoon kuin yksin ohjelmoineet opiskelijat. Samaisessa tutkimuksessa saatiin tuloksia, joiden mukaan pariohjelmoinnista on eniten hyötyä vaikeissa tehtävissä, kun taas helpoissa tehtävissä pariohjelmoinnista ei ole yhtään apua. Salleh ym. (2011) löysivät tutkimuksessaan eriäviä tuloksia kuin edeltävässä Rajalan ym. (2011) tutkimuksessa. Sallehin ym. tutkimuksessa löydettiin tuloksia, joiden mukaan pareittain tehtävissä ohjelmointiharjoituksissa käytetään vähemmän aikaa kuin yksin tehtäessä.

Rajalan ym. (2010) aikaisemman tutkimuksen mukaan, jossa tutkittiin algoritmien oppimista, saatiin lopputuloksena, että pareittain opiskeltaessa ei oppimistuloksissa huomattu mitään eroja verrattuna yksin opiskelleisiin. Tätä tukee myös Sallehin ym. (2011) tutkimus jossa ei löydetty tilastollisesti merkittäviä eroja pariohjelmointia käyttävien opiskelijoiden ja yksin ohjelmoivien välille kurssien lopputuloksissa. Tästä hieman poiketen samassa tutkimuksessa löydettiin, että pariohjelmointia käyttäneet opiskelijat saivat parempia tuloksia kurssitehtävistä kuin yksin ohjelmoineet opiskelijat.

Sallehin ym. (2011) systemaattisessa kirjallisuustutkimuksessa huomattiin, että oppilaat pyrkivät pariutumaan yleensä sellaisen parin kanssa, joka on samalla tasolla heidän kanssaan. Samalla tulokset osoittivat, että oppilaat suoriutuvat paremmin, mikäli pareilla on samanlaiset kyvyt ja yhtäläinen motivaatio kurssin läpäisemiseen. Sukupuolta on myös tutkittu Smithin ym. (2017) tutkimuksessa, jossa tuloksena saatiin, että naisilla pariohjelmointi kurssin aikana oli haitallista verrattaessa kurssin aikana saatuja tuloksia tentistä saatuihin tuloksiin. Miehillä tulokset olivat samansuuntaisia, mutta ero ei ollut yhtä suuri kuin naisilla.

Useat tutkimukset keskittyivät lyhyen aikavälin vaikutuksiin pariohjelmoinnissa, mutta Smithin ym. (2017) tutkimuksessa tutkittiin, kuinka pariohjelmointi vaikutti pitkällä aikavälillä. Heidän tutkimuksessaan tutkittiin kuinka CS2 kurssilla suoritettu pariohjelmointi vaikuttaa CS3 kurssin suoritukseen. Tulokset osoittivat, että pariohjelmointia CS2 kurssilla käyttäneet opiskelijat saivat parempia tuloksia CS3 kurssilla. Tutkimuksessa löydettiin myös CS2 kurssin tuloksista, että pariohjelmoineet opiskelijat saivat projektista ja tehtävistä enemmän pisteitä kuin yksin opiskelleet, mutta tentissä tämä oli kääntynyt päälaelleen, jolloin yksin ohjelmoineet saivat

paremmat pisteet itse tentissä. Eniten pariohjelmointi auttoi heikoimpia opiskelijoita, joiden tulokset kasvoivat eniten pariohjelmoinnin myötä.

Yhteenvetona voitaneen todeta, että pariohjelmoinnista on hyötyä kurssin tehtävien teossa (Seyam ym. 2018), jolloin niistä saatavat pisteet kasvavat. Tentissä pariohjelmointia käyttäneet suoriutuvat hieman heikommin kuin yksin opiskelleet. Tälle ei ole löydetty yhtä tiettyä syytä mutta voidaan arvella, että tämä johtuu siitä, että ne jotka haluavat opiskella yksin ovat jo valmiiksi kyvykkäämpiä kuin pareittain opiskelevat. Parhaan hyödyn pariohjelmoinnista saavat heikoimmat opiskelijat, mutta mikäli tehtävät ovat liian helppoja, ei pariohjelmoinnista ole tällöin hyötyä. Tehtävien tulisi olla riittävän haastavia, jotta pariohjelmoinnista on hyötyä.

2.3.3 Aktiivinen oppiminen

Aktiivisen oppimisen tavoitteena on saada opiskelijat osallistumaan opetukseen enemmän aktiivisesti tekemällä asioita ja osallistumalla enemmän oppimisprosessiin. Pedagogiikka aktiivisen oppimisen taustalla nojaa vahvasti konstruktivistiseen oppimisteoriaan, jossa on tapana, että opiskelijat tekevät yhteistyötä osana tiedon löytämistä ja oppimista (Wulf 2005). Konstruktivismia voidaan pitää objektivistisen oppimisteorian vastakohtana. Objektivistisessa oppimisteoriassa opettajaa pidetään alansa ammattilaisena ja kaikki tieto ns. kaadetaan opettajalta oppilaille. Näin tiedon lähteinä ei ole muita kuin opettaja itse. Konstruktivistisen oppimisteorian mukaan opettaja toimii yhteistyössä oppilaiden kanssa ja auttaa oppilaita ymmärtämään ja rakentamaan tietoa perustuen oppilaiden omiin kokemuksiin. Opiskelijat kehittävät yleensä yhteistyössä muiden oppilaiden kanssa omat tietonsa ja taitonsa, sekä hankkivat käytännön asioista tietoa joko aktiivisessa luokkaympäristössä tai vastaavassa tilassa tai tilanteessa. Suunniteltaessa aktiivisen oppimistyylin käyttöönottoa on hyvä huomioida Bloomin taksonomia (kuva 2), jossa luokitellaan oppimisen lopputulokset hierarkkisesti ulkoa muistamisesta tiedon soveltamiseen ja uuden luomiseen (Wulf 2005).

Kuva 2 Bloomin taksonomia (Lähde https://lehtoripentikainen.wordpress.com/arviointiperusteet/bloomin-taksonomia/) 4/2018

Aktiivista oppimista ja luokkahuonetta on tutkittu paljon. S.Freeman ym. (2014) meta-analysoivat 225 tutkimusta koskien tieteen, teknologian, matematiikan ja insinöörien kursseja, joissa raportoitiin koetuloksia tai epäonnistumisprosentteja, verraten perinteistä luento-opetusta ja aktiivista opetusta. Tulokset osoittivat, että keskimääräiset tenttitulokset nousivat 6% käytettäessä aktiivista oppimista ja oli 1.5 kertaa todennäköisempää, että opiskelija luovuttaa kesken kurssin käytettäessä perinteistä luento-opetusta verrattaessa aktiiviseen opetukseen. Tulokset osoittivat myös, että aktiivinen opetus toimii kaikilla luokkakoilla, vaikka suurimmat parannukset tuloksissa tulivat alle 50 oppilaan kursseista.

Jotta aktiivinen oppiminen olisi mahdollisimman tehokasta, tulee luennot ja kurssi suunnitella etukäteen tarkasti miettien. Pelkän perinteisen luento-opetuksen ja aktiivisen oppimisen yhdistäminen ei suurella todennäköisyydellä tule onnistumaan (Seeling&Eickholt 2017). Opettajien joukossa on oltu huolissaan, että osaavatko he toteuttaa aktiivista opetusta oikeilla tavoilla (Gunn&Raven 2017). Tämä pystytään ratkaisemaan kouluttamalla opettajia aktiivisen opettamisen tapoihin ja ideoihin, jotta nämä pystyisivät paremmin omaksumaan aktiivisen opettamisen periaatteet. Muita haasteita aktiivisessa oppimisessa on hiljaisten ja/tai ei aktiivisten oppilaiden mukaan saaminen opetukseen, ja heidän valmistamisensa luennoilla odottaviin tehtäviin (Yonglei&Nandigam 2016). Muita haasteita on luokkahuoneiden/opetusympäristön muokkaaminen tukemaan aktiivista oppimista. Esimerkiksi isoissa luentosaleissa penkit ovat usein tiukasti kiinnitettyinä lattiaan jolloin näiden tilojen muokkaaminen tukemaan aktiivista

ryhmätyöskentelyä saattaa olla erittäinkin haastavaa ja evätä luennoitsijoiden halua muuttaa opetustyyliään (Gunn&Raven 2017).

Haasteista huolimatta usean tutkimuksen mukaan ensimmäisten yliopiston ohjelmointikurssien muuttamisesta noudattamaan edes osittain aktiivista oppimista, on ollut hyötyä sekä oppilaiden tyytyväisyydessä, että oppimistuloksissa (Hayat ym. 2017, Lokkila 2017, Yonglei&Nandigam 2016).

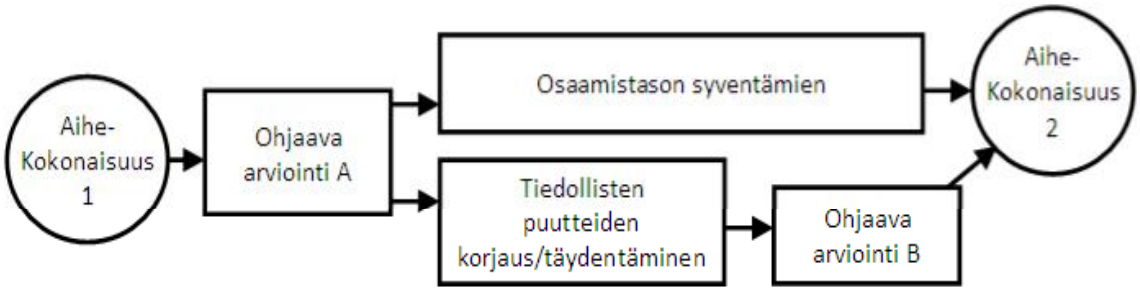
2.3.4 Mastery learning

Mastery learning on Bloomin kehittämä opetusmetodi (Guskey 2007), jossa tietoa jaetaan normaalia pienemmissä aihekokonaisuuksissa. Opiskelijoille annetaan aina yhden aihekokonaisuuden jälkeen mahdollisuus osoittaa opetetun aihealueen hallinta (mastery) ja heille annetaan tämän jälkeen palautetta tuloksistaan. Mikäli aihealueen hallinta ei ollut riittävää niin opiskelijalle annetaan tarkkoja ohjeita siitä, mitä tarvitsee vielä opiskella. Kyseistä sykliä voidaan toistaa teoriassa niin monta kertaa kuin on tarvetta.

Bloom perustaa mastery learning oppimisteoriansa kahteen ajatukseen (Bekki ym. 2012):

1) Toteuttaa John Carrolin teorian siitä, että opiskelijan kyvykkyydellä mitataan enemmän tahtia, jossa aiheita pystytään omaksumaan. Sen sijaan että se mittaisi kykyä omaksua tai oppia aihealueita.
2) Ideaali oppimisympäristö on sellainen, jossa yhtä oppilasta kohden on yksi opettaja.

Kyseisellä opetusmetodilla pyritään varmistamaan jokaisen oppilaan tasapuolinen kohtelu ja annetaan opiskelijoille pienempiä aihekokonaisuuksia käsiteltäväksi. Kyseisten aihekokonaisuuksien kanssa opiskelija pääsee siirtymään seuraavaan aihekokonaisuuteen (kuva 3.) vasta kun tämä on hallinnut edellisen kokonaisuuden. Tarvittaessa annetaan lisäohjeita opiskelijalle. Näin pystytään ehkäisemään virhekäsityksiä sekä mahdollista tiedollisten puutteiden kasautumista isommaksi ongelmaksi.



Kuva 3 Mastery learning käytännössä (Lähde http://maot.fi/oppimisymparisto/mastery-learning/) 4/2018

Mccane ym. (2017) tutkivat kuinka mastery learning vaikuttaa ohjelmoinnin opetukseen. Tutkimuksessaan Mccane ym. toteuttivat ensimmäisen periodin valinnaisen python kurssin ennen toisen periodin pakollista Java-kurssia käyttäen mastery learning menetelmää. Kyseinen python kurssi oli järjestetty perinteisin menetelmin 2009-2014, vaatimattomin tuloksin. Vuonna 2014 kurssi muutettiin noudattamaan mastery learning metodia, koska aiemmista kursseista ei ollut hyötyä pakollisella Java-kurssilla. Python kurssille luotiin 7 välitestiä, jotka julkaistiin 1 testi/viikko tahdilla. Opiskelijat saivat edetä seuraavaan testiin vasta sen jälkeen, kun he olivat läpäisseet edellisen testin. Kurssi koostui lopulta vuonna 2015 podcasteista, lectoriaaleista (opiskelijat pystyivät tulemaan paikanpäälle kysymään luennoitsijalta kysymyksiä), laboratoriotunneista, käytännönkokeesta, 7 välitestistä ja loppukokeesta. Tutkimuksen lopputuloksena todettiin, että heikoimmat opiskelijat hyötyivät kyseisestä opetusinterventiosta suurimmin, ja että aiemmin lähes hyödyttömästä valmistavasta kurssista saatiin hyödyllinen opiskelijoille. Uudistuksen jälkeen etenkin heikoimmat opiskelijat, jotka kävivät valmistavan kurssin, suoriutuivat pakollisen Java-kurssin ensimmäisestä puolikkaasta paremmin effect sizen ollessa 0.6. Mccane ja kumppanit toteavat, että mastery learningistä on ollut hyötyä tässä tapauksessa ja lisäksi oppilaat raportoivat itse parempia oppimistuloksia. Kritiikkiä opetusmetodista annettiin itseopiskelun painottamisesta ja aikarajasta, jossa kurssi tulisi suorittaa (yksi lukukausi).

Vastaavia tuloksia löytyi Kulikin ym. (1990) meta-analyysi tutkimuksessa, jossa tutkittiin mastery learningin vaikutuksia. 103:sta tutkimuksesta joissa raportoitiin tenttituloksia, vain seitsemässä raportoitiin negatiivisia lopputuloksia. 13:sta tutkimuksesta, joissa vertailtiin opiskelijoiden eroja, yhdeksässä tutkimuksessa vähemmän kyvykkäät opiskelijat hyötyivät enemmän mastery learningistä. Muita tuloksia olivat mm. että opiskelijoiden yleiset mielipiteet kurssista paranivat verrattaessa kontrolliryhmiin. Opiskelijoiden asenteiden opetettavaa aihealuetta kohtaan raportoitiin parantuneen 12/14 tutkimuksessa mastery learningin myötä.

Engle&Rollins (2013) tutkivat mastery learningiä hieman eri näkökulmasta, sillä he yhdistivät ohjelmoinnin opetukseensa mastery learningin sekä ammattilaistasoisen koodiarvioinnin. Tutkimuksessaan he muokkasivat ohjelmointikurssia niin, että opiskelijat ohjelmoivat ensin yhden projektin valmiiksi ja tämän ollessa riittävän hyvä pääsivät he seuraavaan projektiin. Kurssin arvostelu riippui suoritettujen projektien määrästä. Yhdistämällä mastery learning ja ammattilaistason koodinarviointia opiskelijat saatiin luomaan parempilaatuista koodia ja monimutkaisten konseptien hallinta parani opiskelijoiden joukossa.

2.4 Opetusteknologiat

Ohjelmoinnin visualisoiminen saattaa olla yksi keino helpottaa ohjelmoinnin opiskelua. Kaila ym. (2009) käyttivät tutkimuksessaan ViLLE-järjestelmää ohjelmoinnin visualisoimiseen ja saivat Napsin ym. (2002) hypoteesin mukaisia tuloksia siitä, että pelkästään ohjelmoinnin visualisointi itsellään ei edesauta oppimista. Ohjelmoinnin visualisoinnista voidaan saada hyötyä, mikäli se yhdistetään aktiiviseen tekemiseen ja oppimiseen. Al-Tahatin ym. (2016) tutkimuksen mukaan naisten asenteet ohjelmointia kohtaan parantuvat käytettäessä ohjelmoinnin visualisointia.

Ohjelmoinnin visualisointi voidaan sisällyttää oppimisjärjestelmään. Oppimisjärjestelmän käyttöä voidaan perustella ohjelmoinnissa mm. suurilla ryhmäkoilla, välittömän palautteen saamisella ja opettajan työtaakan pienentämisellä. Oppimisjärjestelmät auttavat ohjelmoinnin oppimisessa oikein toteutettuina (Kaila ym. 2014; Karavirta ym. 2015; Robinson&Carroll 2017). Kaila ym. (2014) löysivät tutkimuksessaan kolme sääntöä oppimisjärjestelmän käyttöä varten. Ensimmäisenä on esitä ja integroi, jolla tarkoitetaan järjestelmän integroimista tiiviisti kurssin sisältöön ja järjestelmän käyttämistä aktiivisesti kurssin aikana. Toisena sääntönä voidaan pitää opiskelijoiden sitouttamista järjestelmän käyttöön. Tällä tarkoitetaan sitä, että mitä enemmän opiskelijat käyttävät järjestelmää, sitä parempia tuloksia saadaan. Viimeisenä sääntönä pidetään järjestelmän pakollisuutta, mutta kuitenkin opiskelijoiden palkitsemista sen käytöstä. Näin kaikki opiskelijat saadaan käyttämään järjestelmää, vaikka he eivät sitä muuten käyttäisi ja palkitsemisella saadaan opiskelijat motivoitumaan järjestelmän käytöstä.

Muitakin opetusteknologioita voidaan käyttää, kuten Malliarakisin ym. (2016) tutkimuksessa käytetty kouluttava MMORPG peli nimeltään CMX. Tutkimuksessaan he käyttivät kyseistä peliä ensimmäisen vuoden IT opiskelijoilla opettaakseen muutamia proseduraalisen ohjelmoinnin konsepteja. Vertailukohteena ollut opiskelijaryhmä joka ei käyttänyt CMX:ää opiskelussa, sai tutkimuksen mukaan huomattavasti huonompia arvosanoja kuin CMX:ää käyttänyt ryhmä. Tätä voidaan pitää eräänlaisena interventiona joka Vihavaisen ym. (2014) tutkimuksen mukaan voi jo itsessään parantaa oppimistuloksia.

2.5 Opetusmetodit

Seuraavaksi käsiteltävät viisi eri opetusmetodia perustuvat Vihavaisen ym. (2014) tutkimukseen ohjelmoinnin opetuksen interventioista ja niiden tehokkuudesta. Vihavainen ym. jakavat opetusmetodit viiteen eri luokkaan joita ei voida suoraan erotella toisistaan, vaan ne voivat osittain olla myös päällekkäisiä ja niitä voidaan käyttää saman kurssin aikana.

Viisi eri opetusmetodia ja niihin kuuluvat lähestymistavat olivat:

1) yhteistyö ja vertaistuki
    a. yhteistyössä tapahtuva oppiminen, ryhmätyön kautta tapahtuva oppiminen ja pariohjelmointi.
2) alustavat kurssit tai kurssiosiot ennen varsinaista kurssin aloitusta (Bootstrapping)
    a. visuaalisen työkalun käytön sekä CS0 kurssin perustaminen.
3) Kontekstualisointi tai samaistuttavat aiheet
    a. median manipulointi tietokoneen avulla (media computation) ja pelillistäminen.
4) Kurssin muoto, arviointi ja resurssit
    a. luokkakoon muuttaminen, nykyisten resurssien parantaminen ja arviointikriteerien muuttaminen.
5) hybridit
    a. median manipulointi tietokoneen avulla (media computation) pariohjelmointia käyttäen, äärimmäiset oppipoikametodit ja yhteistyössä tapahtuva oppiminen samaistuttavien aiheiden kanssa (kuten pelit).

Kyseisistä opetusmetodeista tehokkaimmaksi osoittautui yhteistyö ja vertaistuki, mutta tilastollisesti merkittäviä eroja ei löytynyt yhdenkään opetusmetodin välille. Jokainen opetusmetodi tuotti parempia tuloksia kuin alun perin käytetty opetustapa, joten täten voitaneen päätellä, että tärkein tekijä ohjelmoinnin opetuksessa on jonkin intervention käyttö. Tiedostettu muutos nykytilanteessa ajaa parempiin tuloksiin.

Yhteistyötä ja vertaistukea on tutkinut mm. Wheeler ym. (2008) wiki -tyyppisen opetusmetodin avulla. Wikiä käytettäessä opiskelija kirjoittaa avoimesti ns. wikiin ja muilla opiskelijoilla on mahdollista arvostella ja muokata wikiin kirjoitettua tekstiä. Wiki osoittaa monia mahdollisuuksia mutta myös monia haasteita. Haasteiden joukossa mm. kritiikin saamisen pelon vähentäminen opiskelijoilla, tekijänoikeusasiat ja kirjoittamisen saaminen yhteistyön suuntaan kilpailun sijaan. Wheeler toteaa kuitenkin wiki -opetusmetodissa piilevän paljon mahdollisuuksia, mutta tätä käyttääkseen tulisi opettajan kiinnittää suurta huomiota itse kurssin toteutukseen. Wheeler

suosittelee wiki -tyyppisen opetuksen käyttöä ja uskoo tämän parantavan oppimistuloksia ja tyytyväisyyttä kurssiin.

Alustavan kurssin luontia on tutkittu Dierbachin ym. (2005) toimesta. Tutkimuksessa perustettiin CS0 kurssi ennen CS1 kurssin aloitusta ja tarkkailtiin, kuinka CS0 kurssin suorittaneet suoriutuvat CS1 kurssista. Tuloksien mukaan CS0 kurssin suorittaneet saivat keskimäärin parempia tuloksia CS1 kurssista kuin kaikki muut opiskelijat. Lisäksi CS0 kurssin suorittaneiden opiskelijoiden tyytyväisyys ja itsetunto vaikuttivat kohonneilta CS1 kurssin aikana.

Kontekstualisointia ovat tutkineet mm. Sprint&Cook (2015) pelillistämisen kautta yliopiston CS1 kurssilla. Heidän tutkimuksensa mukaan pelillistetyn CS1 kurssin todettiin olevan suositumpi opetustapa, verrattaessa perinteiseen luento-opetukseen. Iosup&Epemanin (2014) pelillistämistutkimuksessa löydettiin tuloksia, joiden mukaan luentojen aikainen osallistuminen opetukseen kasvoi, läpäisyprosentit suurenivat ja opiskelijat nauttivat siitä, että kurssin pystyi läpäisemään usealla eri tavalla. Iosup&Epemanin tutkimuksessa koko CS kurssi oli pelillistetty, ja kurssi läpikäytiin ns. pelinä, kurssin opetusmateriaali oli dynaamista ja opiskelijat saivat edetä kurssimateriaalissa omaan tahtiinsa. Kurssin suorituksen arvosanat perustuivat kokonaisuudessaan pelillistetyn kurssin aikana kerätyistä pisteistä, tulostaulukoista, merkeistä, tasoista ja avatusta sisällöstä.

Lähes kaikki tutkimukset osoittivat, että muutettu opetusmetodi paransi lopputuloksia. Nämä tukevat osaltaan olettamusta ja Vihavaisen ym. tutkimusta siitä, että mikä tahansa interventio opetuksessa parantaa lopputuloksia.

3 Opetuksen arviointi

Informaatioteknologian tutkijat raportoivat, että opiskelijoiden tyytyväisyys kasvaa, mikäli heitä haastetaan. Jos opiskelijan kohtaama haaste on liian suuri, niin opiskelijan motivaatio kärsii ja tulokset heikkenevät (Walker 2011; Dorodchi ym. 2017). Kyseisen haasteen saaminen täsmälleen oikeaksi on todella haastavaa. Kuinka on mahdollista mitata haastetta ja testata opiskelijaa? Tämä saattaa riippua opiskelijan tavasta opiskella ja oppia uutta. Erilaiset testaustavat ja kysymystyypit sopivat paremmin osalle opiskelijoista kuin toisille riippuen heidän tavastaan omaksua tietoa (Leithner 2011). Testauksen kognitiivisena taustamateriaalina on usein käytetty Bloomin taksonomiaa tai tästä myöhemmin kehiteltyä muokattua Bloomin taksomiaa, joissa molemmissa oppiminen luokitellaan alhaalta ylöspäin etenevänä kolmiona ja tavoitteena on saada opiskelija niin korkealle tasolle kolmiossa kuin mahdollista (Dorodchi ym. 2017). Seuraavaksi tullaan perehtymään erilaisiin opetuksen arviointitapoihin ja pyritään selvittämään kuinka erilaiset testaustavat eroavat toisistaan.

3.1 Kirjoittamalla tapahtuva arviointi

Kirjoittamalla suoritettu testaus tapahtuu perinteisesti paperille essee kysymyksenä. Tässä tapauksessa kirjoitettuna testauksena tullaan käsittelemään kaikkia tapoja, joissa käytetään kynää ja paperia. Sähköiset tentit käsitellään erikseen myöhemmässä luvussa.

Kirjoitettuja testaustyyppejä voi olla monia, esimerkiksi monivalintakysymykset, essee-kysymykset ja lyhyet selitystehtävät. Näiden kaikkien testaustapojen välillä on eroja riippuen opiskelijasta (Leithner 2011; Simonova&Poulova 2016). Opiskelijoiden tapa opiskella ja heidän käyttämänsä opiskelutavat, sekä opiskeltava ala, saattavat vaikuttaa suuresti siihen, minkä tyylisen tentin opiskelijat haluavat suorittaa. Esimerkiksi kirjallisuuden opiskelijat suosivat todennäköisesti erilaisia testaustapoja kuin kauppatieteiden opiskelijat (Leithner 2011).

Simonova&Poulova (2016) tutkivat kahtena lukuvuonna 2014 ja 2016 kuinka Tšekin yliopiston IT ja hallinto opiskelijat halusivat suorittaa tenttinsä ja miten he toivoivat, että heitä testattaisiin. Lopputuloksena he saivat seuraavat testaustavat, joita kyseiset opiskelijat suosivat kirjoitetuista vaihtoehdoista:

- Monivalinta kysymykset, yksi vaihtoehto on oikein
- Kyllä/ei kysymykset

- Totta vai tarua kysymykset (vuonna 2014)
- Opintojakson kestävä projekti jonka aihe asetetaan kurssin alussa. Lopputulos esitellään kurssin päättyessä (kirjoitettu testi) (vuonna 2016)

Testaustavat joita ei haluttu nähdä olivat:

- kirjoitetaan essee joka vastaa kysymykseen tai ongelmaan, aihe valitaan tuntemattomalta listalta
- Koepäivän aikana opiskelijat suorittavat projektin, jonka aihe annetaan aamulla. Aiheiden listaa ei tiedetä etukäteen.
- Opintojakson kestävä projekti jonka aihe asetetaan kurssin alussa. Lopputulos esitellään kurssin päättyessä (vuonna 2014)
- Totta vai tarua kysymykset (vuonna 2016)

Simonova&Poulova toteavat tutkimuksensa päätteeksi, että he löysivät joitain eroja siinä mitä preferenssejä opiskelijoilla oli testaustavasta, ja että tulevaisuudessa olisi mielenkiintoista tarjota opiskelijoille vaihtoehtoja heidän suorittaessaan tenttiä. Kyseistä aihetta on tutkinut Leihtner vuonna 2011.

Leihtner (2011) tutki mitä vastausvaihtoehtoja opiskelijat valitsevat, mikäli heille annetaan mahdollisuus valita kirjoitetussa tentissä. Tutkimuksen mukaan opiskelijat jotka tuntevat itsensä ja parhaan tyylinsä oppia, osaavat valita itselleen sopivat tehtävät tentissä, mikäli vaihtoehtoja on. Lopputuloksena saatiin, että mikäli tentissä annetaan opiskelijoille mahdollisuus valita mihin kysymyksiin he haluavat vastata, esimerkkeinä essee kysymykset, monivalinta- ja lyhyet selitä tyyliset tehtävät niin opiskelijat pystyvät tällöin valitsemaan itselleen parhaiten sopivan testausmenetelmän ja saavat näin parempia tuloksia. Lehtner toteaa tutkimuksensa perusteella, että oppimistyylien huomioonottaminen testattaessa opittua tietoa on kannattavaa, mutta ei pysty sanomaan tätä varmaksi vain yhden tutkimuksen perusteella.

3.2 Suullisesti tapahtuva arviointi

Suulliseen arviointiin yhdistetään usein muiden opiskelijoiden arviointi, sillä usein testausmenetelmänä on esitys jonka opettaja sekä muut opiskelijat arvioivat. Poikkeuksena voitaneen pitää esimerkiksi kielten opiskelua, jossa suullinen arviointi on todennäköisesti ainoa tapa mitata opiskelijan kykyä puhua opiskelemaansa kieltä. Arviointi voidaan suorittaa joko ryhmätyönä tai yksittäin keskustellen esimerkiksi opettajan kanssa.

Edellisessä luvussa mainitut Simona&Poulova (2016) tutkivat kirjoitettujen testaustapojen lisäksi myös suullisesti suoritettavia testaustapoja. Heidän tutkimuksessaan löydettiin seuraavat suulliset testaustavat joita opiskelijat suosivat:

- Opettaja valitsee kysymyksen ennalta valitusta listasta.
- Kysymys ”Mistä olit eniten kiinnnostunut tässä aiheessa?” joka aloittaa opettaja-opiskelija keskustelun, jossa opettaja kuuntelee ilman keskeytyksiä.
- Kysymys ”Mistä olit eniten kiinnnostunut tässä aiheessa?” joka aloittaa opettaja-opiskelija keskustelun, jossa opettaja esittää lisäkysymyksiä aiheesta.

Testaustavat joita opiskelijat eivät toivoneet käytettävän:

- Opettaja valitsee kysymyksen tuntemattomalta listalta
- Istutaan pyöreän pöydän ääressä, jossa opiskelijat vastaavat samaan kysymykseen lisäten edelliseen vastaukseen aina jotain lisää (nopein vastaaja saa aloittaa). Suoritetaan useita vastauskierroksia.
- Istutaan pyöreän pöydän ääressä, jossa opiskelijat keskittyvät samaan kysymykseen tai ongelmaan ja he käyttävät kysymyksen tai ongelman ratkaisuun kriittistä analysointia, aikaisempaa tietoa ja kokemusta sekä arviointia yms. keinoja.

Suullista testaamista voidaan käyttää myös osana arviointia, kuten testaamaan kirjoitetun ohjelmiston oikeaa ymmärtämistä opiskelijalta (Kohana&Okamoto 2016). Selvi&Chandramohan (2016) tutkimuksessa arvioitiin, kuinka eräänlaisten kaukosäätimien käyttö vaikuttaa oppilaiden antamaan palautteeseen esityksistä. Lopputuloksena saatiin, että opiskelijat pitivät näiden kaukosäätimien käytöstä palautteen antamisessa ja että tärkeänä osana tässä oli anonymiteetti jonka kaukosäätimet takaavat. Tästä voitaneen päätellä, että opiskelijoiden toisilleen antama palaute suullisista esityksistä on helpointa antaa anonyymisti ja vaikuttaakin siltä, että kyseiset kaukosäätimet ovat väistymässä kännyköiden yms. tieltä (Meseguer-Dueñas 2018). Opiskelijoiden toisilleen antama palaute voisi helposti olla puolueellista mutta Meseguer-Dueñas ym. (2018) tutkimuksen mukaan annettaessa riittävän tarkat arvosteluasteikot (rubrics) ja tarvittaessa ohjeita opettajalta, akateemisessa ympäristössä suoritettavien suullisten esitysten arvostelu muiden opiskelijaryhmien toimesta on samalla tasolla luennoitsijoiden antamien palautteiden kanssa. Mikäli käytetään opiskelijoiden antamaa palautetta, on akateemisessa ympäristössä suositeltavaa yleensä käyttää pieniä ryhmiä, yhteisesti sovittuja ja hyväksyttyjä kriteerejä ja arvioida kokonaisuutta (Falchikov&Goldfinch 2000).

Suullisen arvioinnin työkaluna käytetään siis usein muita opiskelijoita ja tämä todennäköisesti onnistuu käytettäessä oikein. Opiskelijoiden toisilleen antama palaute on yhtä validia kuin

luennoitsijoiden antama palaute. Tärkeässä osassa onkin palautteen saanti ja näin opiskelija pystyy kehittymään edelleen (Meseguer-Dueñas ym. 2018).

3.3 Ryhmätyönä tapahtuva arviointi

Ryhmätyötä on pitkään pidetty tärkeänä testaustapana ja sen käyttöä on perusteltu seuraavan viiden hyödyn mukaan (Davies 2009):

- Ryhmätyö edistää ”syvä” oppimista.
- Ryhmätyö edistää aktiivista oppimista.
- Ryhmätyö korostaa kokeilevaa, yhteistyössä ja yhteisesti suoritettavaa oppimista.
- Ryhmätyötä voidaan perustella sillä, että se edistää tiedon rakentumista ja parantaa ongelmapohjaista oppimista.
- Ryhmätyön on myös sanottu olevan autenttinen arvioinnin muoto otettaessa huomioon opiskelijan tuleva työllistyminen. Ryhmissä työskenteleminen on tärkeä osa henkilökohtaista uraa ja rekrytoijat kysyvät opiskelijoilta usein heidän työkokemustaan ryhmätoiminnasta.

Ryhmätyön arvioimisen ongelmaksi muodostuu, kuinka määritellä yksittäisen opiskelijan suoritus ryhmässä. Kokonaiselle ryhmälle annettavaa arvosanaa tulisi välttää, sillä tämä ei motivoi opiskelijoita, luo esteitä yhteistyölle, saattaa luoda kitkaa ryhmän jäsenten välille ja kyseiset arvosanat ovat epäreiluja King&Behnke (2005). Ongelmaksi muodostuvat vapaamatkustajat ja ns. sucker effect vaikutus. Tällä tarkoitetaan sitä, että opiskelija joka huomaa muun ryhmäläisen olevan vapaamatkustaja, ja kokee tämän epäoikeudenmukaiseksi, laskee omaa panostaan ryhmätyöhön ja näin kyseisestä opiskelijasta saattaa muodostua toinen vapaamatkustaja Davies (2009).

Kyseisiä ongelmia ratkaisemaan on kehitetty mm. pelillistämiseen perustuva arviointityökalu. Kyseisessä tavassa opiskelijat suorittavat ryhmätyötä internetin välityksellä jaetuissa työtiloissa, jossa jokaisesta osallistumisesta (esimerkiksi lähteen linkkaamisesta, lähteen lukemisesta, kommentoinnista, muiden ohjeistamisesta yms.) saa pisteitä. Opiskelijat eivät tiedä pisteitään tai paljonko pisteitä toiminnoista saa. Opiskelijat tietävät ainoastaan, että toiminnoista saa pisteitä. Näiden pisteiden avulla opettaja pystyy kurssin päätteeksi arvioimaan pisteiden perusteella jokaisen ryhmäläisen arvosanan (Moccozet 2013).

Ryhmätyötä voidaan arvioida monin eri tavoin, mutta jokaisessa arvostelutavassa tulee aina ottaa huomioon mitä ryhmätyötä ollaan tekemässä, millaisia tehtäviä ratkotaan ja arvosteluasteikko sekä ryhmän koko joka Daviesin (2009) tutkimusten mukaan tulisi maksimissaan olla 3-4. Tätä suuremmissa ryhmissä yksittäisen ryhmäläisen antama työpanos pienentyy ja vaara vapaamatkustajista ja ns. social loafingista kasvaa.

3.4. Vertaisarviointi

Vertaisarvioinnissa arvosanat perustuvat usein muiden ryhmäläisten tai vertaisten antamiin arvosteluihin. Ongelmia tässä tavassa ovat mm. ylimääräinen työmäärä jonka tämä tuo opiskelijoille ja henkilökohtaiset suhteet opiskelijoiden välillä, jotka saattavat vaikuttaa annettuihin arvosanoihin ja näin vääristää lukemia. Patil ym. (2016) pyrkivät löytämään ratkaisun jälkimmäiseen ongelmaan. He kehittivät järjestelmän nimeltä ”luotettava suhteellinen vertaisarviointi” (”Trusted relative peer review”) jossa yhdistetään kolmea eri vertaisarviointitapaa ja näistä saatujen tulosten perusteella laskettiin suhteellinen arvosana, josta saatiin lopulta lopullinen arvosana. Kolme käytettyä vertaisarviointitapaa olivat Michaelsen, Fink ja Kolesin metodit. Lopputuloksena saatiin, että kyseisestä tavasta on hyötyä ryhmän jäsenten arvostelussa, arvosanoille saatiin suurempi jakauma kuin perinteisessä vertaisarvioinnissa ja oppilaat pitivät pääosin kyseistä arviointitapaa reiluna. Huonoina puolina mainittiin suuri tiedon määrä ja sen käsittelyyn vaadittava suurehko työmäärä.

Mikäli arviointitapana on vertaisarviointi, suositellaan noudatettavaksi seuraavia ohjeita (Falchikov&Goldfinch 2000):

1) Vältetään isoja ryhmiä vertaisarviointia käytettäessä.
2) Suoritetaan vertaisarvioinnin opiskelua perinteisessä akateemisessa ympäristössä ja osallistetaan opiskelijat akateemisten tuotteiden ja prosessien vertaisarviointiin.
3) Ei odoteta opiskelija-arvioijan arvioivan useampia ulottuvuuksia. On järkevämpää käyttää kokonaisvaltaista arviointia hyvin ymmärrettyjen kriteerien kanssa.
4) Osallistetaan opiskelijoita keskustelemaan arviointikriteereistä.
5) Kiinnitetään suurta huomiota opiskeltavan aiheen suunnitteluun, implementaatioon ja raportointiin
6) Vertaisarviointia voidaan käyttää missä tahansa alalla ja millä tahansa tasolla.
7) Vältetään vertaisryhmien ja opettajien välisten sopimussuhteiden käyttöä pätevyyden mittaamisessa.

Vertaisarviointia käytetään usein ryhmätöiden arvioinnissa, jotta opettajat saisivat käsityksen, miten työnmäärä on jakautunut ryhmäläisten välillä ja paljonko kukin ryhmän jäsen on antanut ryhmätyölle. Vertaisarvioinnin ongelmat ovat opiskelijoiden välisissä suhteissa ja siinä, kuinka nämä suhteet otetaan huomioon arvioinnissa. Työkalujen avulla tätä voidaan osaltaan kiertää ja hyvällä suunnittelulla sekä oikeilla arviointikriteereillä vertaisarvioinnista saadaan parempia tuloksia.

4 Sähköinen tentti

Noin 91% 16-24 vuotiaista käyttää tietokonetta (Andreatos 2009). Kyseinen tutkimus on vuodelta 2009, joten voitaneen arvella, että kyseinen luku on vain kasvanut silloisesta. Tämä tarkoittaa sitä, että lähes kaikki opiskelijat käyttävät tänä päivänä tietokonetta tullessaan yliopistoon. IT:n kasvava suosio on johtanut oppilaitokset kehittämään työkaluja, jotka hyväksikäyttävät IT:tä. Yhtenä näistä voidaan pitää sähköistä tenttiä.

Sähköisen tentin arviointityypit ovat Crispin (2011) mukaan:

- diagnostinen arviointi
    - arviointi kurssin alussa, jotta tiedetään opiskelijoiden lähtötaso sekä tiedot
- formatiivinen arviointi
    - arviointi kurssin aikana, jotta nähdään jo opitut asiat ja mitä tarvitsee vielä käydä läpi
- loppu (summative) arviointi
    - kurssin lopussa tapahtuva arviointi, jolla saadaan opiskelijoiden loppuarvosana
- integroiva arviointi
    - Tarkoitus vaikuttaa opiskelijan tulevaan oppimiseen perustuen palautteeseen opettajalta, muilta opiskelijoilta tai itsearvioinnin kautta

Sähköisensä tenttinä käsitellään tässä tutkielmassa sellaista tenttiä, jossa koe tehdään pääasiallisesti tietokoneella ja on laadultaan loppu (summative) tentti. Sähköisiä tenttejä käytetään yhä kiihtyvässä tahdissa yliopistoissa ja muissa oppilaitoksissa. Sähköisen tentin etuina voidaan pitää useaa eri asiaa (Kuikka ym. 2014):

- arviointi voi IT:n myötä olla opiskelijakeskeisempää
- vähentää kokeiden korjaamiseen menevää aikaa
- antaa enemmän mahdollisuuksia arviointiin
- oppilaat sitä mieltä, että parantaa oppimista
- antaa enemmän vaihtoehtoja tentin tehtäviin

Sähköinen koe voidaan järjestää esimerkiksi, kuten Turun yliopistossa on järjestetty: opiskelija varaa ajan tenttiakvaariosta valitsemalleen ajalle ja varatun ajan koittaessa kirjautuu sisään kameravalvottuun tenttiakvaarioon opiskelijakorttinsa kanssa. Kokeen jälkeen opiskelija saa tulokset tehtävistä, jotka voidaan tällä hetkellä arvioida automaattisesti mm. monivalinta-kysymykset. Mm. essee kysymyksien arviointi jää opettajien tehtäväksi ja näin opiskelija saa

lopullisen tuloksen opettajan arvioitua tarvitut kysymykset. Yliopistoissa ja muissa akateemisissa laitoksissa tenttiakvaarioita käytetään useimmissa tapauksissa sähköisten kokeiden suorituksissa.

Opettajat toivovat Kuikan ym. (2014) tutkimuksen mukaan sähköiseltä tentiltä ja järjestelmältä jossa kyseinen tentti pidetään yhdeksää eri asiaa: 1) tuki ja ohjaus järjestelmän käytössä. Kyseisellä tuella tarkoitetaan sitä, että ohjausta on saatavilla niin teknologisiin kuin pedagogisiin ongelmiin. 2) Yksinkertainen käyttöliittymä on tärkeässä osassa, sillä mikäli järjestelmän käyttö on liian vaikeaa, niin tällöin riski järjestelmän käyttämättömyydestä kasvaa. Helppous kokonaisuutena on suuressa osassa, sillä seuraavana toiveena on 3) normaalien ja jo olemassa olevien tunnusten käyttö. Näin opettajan ei tarvitse hankkia uusia tunnuksia sähköisiä tenttejä varten. 4) Automaattinen arviointi ja 5) välitön palaute kuuluvat myös osaksi opettajien toiveita siksi, että molemmat vähentävät opettajan työtaakkaa. Automaattisen arvioinnin piiriin toivottiin kuuluvan kaikki kokeen tehtävät ja välitön palaute tulisi saada välittömästi kokeen jälkeen, sillä tämän on todettu edistävän oppimista. 6) Etsi ja jako toiminto jolla tarkoitetaan sitä, että opettaja pystyy jakamaan tekemänsä koekysymykset muille opettajille ja mahdollisesti saamaan palautetta tehtävistä. 7) Varauskalenteri kokeille, jotta opiskelijat pystyvät varaamaan koeajan ilman opettajan osallistumista varauksen tekoon. 8) Tilastot ja statistiikat kokeista mukaan lukien mm. opiskelijan kokeeseen käyttämä aika ja yritysten määrä. Yhdeksäntenä toiveena löydettiin 9) mahdollisuus liittää materiaalia kokeeseen, mm. video- ja kuvamateriaalia, joka mahdollistaa monimuotoisempien koekysymysten luonnin.

Ongelmia sähköisissä kokeissa ovat tunnistautuminen ja kokeen turvallisuus. Huijaamisen esto liittyy kiinteästi sähköisiin tentteihin, tätä varten Bazar (2017) kehitti algoritmin joka estää kahden samanlaisen kokeen ilmestymisen vierekkäisille kokeen tekijöille. Bazarin tutkimuksen mukaan kyseinen algoritmi toimii ja näin vältetään kokeen aikainen lunttaaminen. Muita turvallisuusseikkoja ovat infrastruktuuriin liittyvät seikat, kuten wlan-ympäristössä järjestetty koe, jota varten on kehitetty mm. yksi malli (kts. Sukadarmika ym. 2016). Huijaamisella voidaan tarkoittaa myös kokeen arvostelijoissa tapahtuvaa vilppiä, esimerkiksi tietyn oppilaan suosiminen tai motivaatiota kajota tuloksiin. Tätä varten on kehitetty monia eri ratkaisuja (kts. esim. Giustolisi 2013; Dreier ym. 2014). Molemmissa tutkimuksissa painotetaan kokeen tekijän anonyymiutta, jonka kautta pystytään vähentämään riskiä huijaamiseen.

Toisena tärkeänä osa-alueena voidaan pitää sähköisen koejärjestelmän toimivuutta. Mikäli ensimmäinen kokemus sähköisestä kokeesta on negatiivinen, todennäköisyys siihen, että sähköistä koetta käytetään uudelleen, pienenee huomattavasti (Wibowo ym. 2016). Katzlinger&Stabauer (2017) toteuttivat tutkimustaan varten sähköisen tenttitilan vuonna 2011 Johannes Keplerin yliopistoon Itävallan Linzissä. He tutkivat kuinka tenttitila oli onnistunut, ja

mitä onnistuneeseen tenttitilaan vaaditaan. Heidän toteutuksessaan tenttitilassa on paikalla yksi valvoja ja mahdollisesti tekninen tukihenkilö joka auttaa tarvittaessa. Tunnistautumiseen käytettiin erillistä korttia ja digitaalista allekirjoitusta. He toteavat tutkimuksensa päätteeksi, että opiskelijat ja opettajat pitivät suuresti kyseisestä toteutuksesta ja myöhemmin samainen toteutus vietiin toiseen itävaltalaiseen yliopistoon. Onnistumisen taustalla olivat hyvin toteutettu ja toimiva järjestelmä sekä hyvät ohjeet ennen kokeen suoritusta.

Sähköisen kokeen kuten perinteisenkin kokeen ongelmana voidaan pitää myös riittävän laadukkaiden kysymysten luontia, sekä mahdollisten kysymyspankkien käyttöehtoja. Mikäli käytössä on oppimisjärjestelmä, on usein myös käytössä jonkinlainen kysymyspooli josta voi valita esimerkiksi muiden opettajien tekemiä kysymyksiä. Ongelmia saattaa syntyä tekijänoikeuskysymyksissä, sekä siinä, mikäli kysymyksiä tai tehtäviä pyritään siirtämään toiseen järjestelmään. Mikäli järjestelmät eivät ole yhteensopivia toistensa kanssa, niin tällöin kysymykset ovat rajattuina vain omiin järjestelmiinsä, mikä osaltaan hidastaa ja haittaa sähköisten kokeiden käyttöönottoa sekä saattaa lisätä opettajien haluttomuutta siirtyä käyttämään sähköisiä kokeita (Brink&Lautenbach 2011).

Oppilaiden puolelta katsottuna sähköiset kokeet tuovat lisää mahdollisuuksia tenttimiseen. Riippuen tietysti sähköisen kokeen toteutustavasta, mutta pääosin etuja ovat mm. tenttikirjallisuuden tasaisempi jakautuminen tenttiaikojen tasaisemman jakautumisen vuoksi. Tämä saattaa osaltaan myös nopeuttaa valmistumista. Mahdollisuus tekstinkäsittelyyn, sekä yleinen tietokoneiden käyttö helpottavat oppilaiden kokeen suoritusta. Tämä saattaa osaltaan myös poistaa oppilaan käsialaan perustuvan ennakkoasenteen opettajan puolesta, mikä saattaisi olla mahdollista käsintehdyissä tenteissä (Thomas ym. 2002).

Tärkeänä voidaan pitää Brink&Lautenbachin (2011) tutkimuksen mukaan myös sitä, että opiskelijat on tutustutettu käytettävään koejärjestelmään jo etukäteen. Näin järjestelmän käyttö etukäteen vähentää kokeen aikana järjestelmästä johtuvaa stressiä, ja opiskelija pystyy keskittymään kysymyksiin järjestelmän käytön sijaan.

4.1. Sähköinen tentti ohjelmoinnissa

Useat opiskelijat jotka läpäisevät ohjelmointikurssit, eivät osaa ohjelmoida (McCracken ym. 2001). Tämä saattaa johtua osaltaan ns. ulkoa oppimisesta. Tällä tarkoitetaan sitä, kuten opiskelija sanoi Traynorin ym. (2006) tutkimuksessa: ”suurin osa koekysymyksistä etsii samaa asiaa ja yleensä pisteitä saa siitä, että vastaus näyttää oikealta. Esimerkiksi jos kysymys koskee

hakutoimintoa, riittää, että vastaukseen laittaa loop rakenteen, array objektin sekä if -lausekkeen niin tällöin saa pisteitä. Ei täysiä pisteitä, mutta riittävästi läpäisyyn.".

Ainakin osittaisena ratkaisuna tähän ongelmaan voidaan pitää Sheardin ym. (2015) tutkimuksen mukaan oikein rakennettuja koekysymyksiä. He tutkivat kuinka muodostaa kysymyksiä ohjelmointikokeisiin ja päätyivät 10 kysymykseen ja ehdotukseen:

1) Pidä kysymykset niin yksinkertaisina kuin mahdollista
2) Harkitse kysymysten kontekstualisointia (ei suositella käytettäväksi)
3) Käytä diagrammeja ja kuvia auttamaan opiskelijaa ymmärtämään kysymys
4) Varmista että kysymystyypit ovat entuudestaan tuttuja opiskelijoille
5) Jos kysymykseen sisältyy koodia, varmista, että se on kirjoitettu, kuten kurssilla on opetettu
6) Vältä tarkoituksenmukaista monimutkaistamista
7) Sisällytä kaiken tasoisia kysymyksiä, helposta-vaikeaan
8) Harkitse monivalinta kysymysten sisällyttämistä kokeeseen
9) Harkitse koodin luku tehtävien sisällyttämistä kokeeseen
10) Sisällytä kysymyksiä joihin vastataan eri tavoilla, valitse kysymyksen vastaustapa sen mukaan mitä halutaan mitata tai testata.

Simonin ym. (2012) tutkimuksessa todetaan, että eri yliopistojen ja ohjelmointikurssien pitäjien väliltä oli vaikea löytää yhteistä mielipidettä siitä, mitä kysymystä voidaan pitää helppona ja mitä vaikeana. Tämän todettiin johtuvan tutkimuksen kontekstista, jossa tutkittiin kurssien loppukoetta, eikä kurssien aikana tapahtunutta opetusta otettu huomioon. Tähän perustuen tutkimuksessa todetaan, että kokeen tehtävät riippuvat aina opetetusta kontekstista ja kurssin pitäjällä on yleensä paras käsitys siitä, mitä on opetettu ja miten tätä tulisi testata. Kts. Anthonyn (2017) tutkimus kysymyksistä, jotka voisivat sopia lähes jokaiseen ohjelmointikokeeseen.

Ohjelmointi tapahtuu luonnostaan tietokoneella. Ohjelmoinnin kokeet voidaan suorittaa myös paperille, mutta tämä ei tue ohjelmoinnin opetusta siinä, että kokeen tulisi muistuttaa oikeaa tilannetta niin paljon kuin mahdollista. Tästä syystä ohjelmoinnin kokeita suoritetaan pääosin tietokoneella erinäisillä ohjelmilla. Usein ohjelmoinnin kursseilla ohjelmointi tapahtuu tietyllä alustalla, esimerkiksi Turun yliopistossa käytössä olevalla ViLLE -järjestelmällä. Kokeen suorittaminen samalla alustalla olisi suotavaa sen takia, että opiskelijat ovat jo kurssin aikana tutustuneet järjestelmään, ja tästä ei koidu ongelmia kokeen suorituksessa. Lisäksi kokeen suorittaminen sähköisessä muodossa mahdollistaa automaattisen arvioinnin. Esimerkiksi Rajalan ym. (2016) tutkimuksessa tutkittiin, kuinka automaattinen arviointi auttaa ohjelmoinnin kokeissa ja kuinka tehtävät mittaavat opiskelijoiden kykyjä. Koe suoritettiin Turun yliopistossa ViLLE-

järjestelmässä, joka arvioi automaattisesti kaikki kyseisen kokeen tehtävät. Vastauksia kokeessa oli yli 5000 kappaletta, joten kokeen arviointi tai suorittaminen samassa mittakaavassa tai samoilla tehtävillä, ilman automaattista arviointia, olisi lähes mahdotonta. Tuloksien mukaan opiskelijat pitivät sähköisestä kokeesta ja kyseistä koetta voitiin pitää onnistuneena esimerkkinä ohjelmoinnin sähköisestä kokeesta. Muita toteutuksia on ollut mm. Richter&Boehringerin (2014) tutkimuksessa käytetty konfiguraatio, jossa koe tehtiin internet ympäristössä, mutta oppilaiden pääsyä muihin ominaisuuksiin rajoitettiin. Kyseisestä tutkimuksesta saatiin hyviä tuloksia ja tämä tutkimus puoltaa jo aiemmin todettua Rajalan ym. tutkimusta siinä, että sähköisiä ohjelmoinnin kokeita voidaan käyttää lopputestausmenetelmänä korvaamaan kynän ja paperin.

Yksi tärkeistä osa-alueista pohdittaessa ohjelmoinnin tenttiä on se mitä halutaan testata. Esimerkiksi hyvät debuggaajat ovat usein hyviä ohjelmoijia. Toisinpäin mentäessä vain noin puolet hyvistä ohjelmoijista ovat hyviä debuggaajia (Ahmadzadeh ym. 2014). Tämä osaltaan kertoo kuinka testaustavan valinta vaikuttaa saatuihin lopputuloksiin. Muita asioita joita tulisi pohtia ohjelmoinnin tenttiä mietittäessä on ohjelmointikielen valinta.

Valittu ohjelmointikieli vaikuttaa suuresti ohjelmoinnin opetukseen ja tenttimiseen. Aikaisemmin käytössä olivat mm. (osa myös edelleen käytössä osittain tai muunneltuna) ohjelmointikielet fortran, cobol ja basic. Tästä siirryttiin hiljalleen pascal ohjelmointikielen käyttöön, kunnes siirryttiin yhä eteenpäin C++ kieleen. C++ otettiin pääosin käyttöön olio-ohjelmoinnin myötä ja tämän rinnalle on sittemmin otettu Java ohjelmointikieli. Syitä sille miksi Java otettiin käyttöön, perusteltiin Javan aidon olio-ohjelmoinnin takia. Nykyisin suurin osa ohjelmoinnin kursseista opetetaan joko Java tai C++ kielellä (Giangrande 2007). Giangranden (2007) mukaan Javan käyttö ensimmäisillä ohjelmointikursseilla on turvallisempaa ja graafisen käyttöliittymän kehittämistä tuetaan suoraan standardisoiduista API:sta. C++ puolestaan antaa opettajan päättää milloin olio-ohjelmointi esitellään kurssilla. Javassa kyseistä vaihtoehtoa ei ole vaan oliot tulee esitellä heti kurssin alussa. Muiksi opetuskieli vaihtoehdoiksi Giangrande mainitsee C# sekä visual basicin. Oli opetuskielenä mikä tahansa, niin tentin automaattinen arviointi on lähes aina mahdollista oikealla järjestelmällä.

Ohjelmoinnin tentin tekeminen paperille olisi nykypäivänä melko vaivalloista. Paperille tehtävässä tentissä ei olisi mahdollisuutta iteroida ja debugata koodattua aineistoa sekä koodin suoritus olisi käytännössä mahdotonta. Lisäksi paperille tehtävä tentti ei vastaisi oikeaa ohjelmointitilannetta lähes millään tavalla. Tämän takia ohjelmoinnin tentit ovat siirtyneet pääosin sähköiseen muotoon. Turun yliopiston jo aiemmin mainittua ViLLE - oppimisjärjestelmää, on käytetty paljon sähköisissä ohjelmoinnin tenteissä, etenkin ensimmäisissä ohjelmoinnin kursseissa. ViLLEn sähköisiä tenttejä on tutkinut Rajala ym. (2016).

Rajalan ym. tutkimuksessa todettiin, että ohjelmoinnin tentissä ohjelmointitehtävät vaativat enemmän aikaa ja keskittymistä kuin muun tyyppiset tehtävät. Yleisesti palautusmäärät ovat korkeammat ohjelmointitehtävissä sen takia, että niistä opiskelijat saavat ViLLE -järjestelmässä palautetta jokaisen palautuksen jälkeen välittömästi. Muissa tehtävissä, kuten monivalinnassa ja koodinjärjestelytehtävissä palautusmäärät olivat huomattavasti pienempiä, sillä näistä tehtävistä opiskelija ei saanut palautetta. Näin Rajalan tutkimuksessa päädytään siihen lopputulokseen, että opiskelijan näkökulmasta mahdollisuus testata ja lähettää tehtävä uudelleen olivat hyödyllisimmät asiat sähköisessä tentissä. Opettajan näkökulmasta katsottuna automaattisesti arvioitavat monivalintatehtävät ja koodinjärjestelytehtävät helpottavat työtaakkaa, mutta eivät anna opiskelijoille samanlaisia hyötyjä kuin ohjelmointitehtävät.

Vaikka käytössä on sähköinen tentti ja -järjestelmä niin opiskelijoiden oppimistavat ja tavat tehdä tenttiä eroavat silti suuresti. Kaiken tasoiset opiskelijat tekevät virheitä ohjelmointitenteissä. Hyvät opiskelijat eroavat huonoista opiskelijoista siinä, että hyvät opiskelijat pystyvät ohjelmointitehtävissä korjaamaan tentin vastauksiaan ja väärinymmärryksiään perustuen järjestelmän antamaan palautteeseen (Kurvinen ym. 2016).

Ohjelmoinnin tentin erityisominaisuutena voitaneen pitää mahdollisuutta vastata tehtävään monella eri tavalla oikein. Tämä siis tarkoittaa sitä, että koodia voidaan kirjoittaa monella eri tavalla ja silti on mahdollista saada sama vastaus. Tätä on tutkittu mm. Españan ym. (2017) toimesta tutkimuksessa, jossa päädyttiin menetelmään ja järjestykseen jossa koodaustehtävä tulisi arvioida.

Ohjelmoinnin sähköisessä tentissä tulee ottaa huomioon usea eri asia kuten mm. seuraavat asiat: mitä järjestelmää käytetään, mikä ohjelmointikieli on käytössä, pystyykö järjestelmä automaattiseen arviointiin vai korjataanko tehtävät käsin, kuinka tehtävät arvostellaan, mitä ja millaisia kysymyksiä tenttiin luodaan, ajankäytön hallinta, palautteen anto tentin aikana ja mitä halutaan testata. Onnistuneen sähköisen tentin valinta on monen asian summa mutta ei mahdotonta. Seuraavaksi tulevassa tutkimuksessa perehdytäänkin mm. kysymykseen: onko Turun yliopiston ensimmäisen ohjelmointikurssin tentti onnistunut?

5 Metodit ja materiaalit

Tutkimuksessa käytetty materiaali on kerätty Turun yliopiston ViLLE -järjestelmällä suoritetusta Algoritmien ohjelmoinnin peruskurssista (tästä eteenpäin Aop). Materiaali koostuu kurssin aikana tehdyistä harjoitteista ViLLE -järjestelmässä (taulukko 1) sekä kahdesta lopputentistä, jotka suoritettiin myös ViLLE -järjestelmässä. Datan on kerännyt ViLLE -team ja se on luovutettu käyttööni anonymisoituna. Aineiston N=273, josta karsin pois ne jotka eivät suorittaneet kurssin tenttiä (89kpl), tai jotka suorittivat vain tentin (2kpl). Karsinnan jälkeen N=184, jota käytettiin lopullisessa kurssiaktiivisuuden tutkimuksessa. Materiaali on pääasiassa kvantitatiivista, lukuun ottamatta tenttipalautteen muutamaa kysymysosiota. Materiaalista on pyritty etsimään, korrelaatiota oppilaan kurssiaktiivisuuden ja tentin lopputulosten välillä. Lopputentin oikeellisuutta ja tarvetta pyritään tarkastelemaan tenttipalautteiden, ja sen perusteella, löytyykö korrelaatiota hyvän kurssiaktiivisuuden ja tentistä saatavien pisteiden perusteella. Lisäksi pyritään analysoimaan kurssin tenttiä perustuen kahdesta lopputentistä kerättyyn dataan (taulukko 2) sekä tenttipalautteiden perusteella. Kahden tentin yhdistetty N=167 joista tenttipalautetta antoi N=149 henkeä.

Kurssin aikana kerätty data N=184	Maksimipisteet
Johdanto	20
7kpl ViLLE harjoituksia (1/vko)	27-52 (riippuen päivästä) yht. max. 322
Luentoläsnäolot	100
Tutoriaaliläsnäolot	7
Demonstraatiot 4krt	400
Pythonista Javaan	110
Muuttujat, merkkijonot ja ehtolauseet	95
Silmukat	100
Metodit	86
Taulukot	96
Valmiiden luokkien käyttö	100
Kertaus	75
Harjoitustyöt 4kpl	30 tai 60 yht. max.180
Kyselyt	50
Harjoitustentti	50
Tenttitulokset	90
Tenttipalaute	5
Kokonaispistemäärä/kurssiaktiivisuus	1560

Taulukko 1 Kurssin aikana kerätty data

Tenteistä kerätty data N=167	Maksimipisteet
Kysymyksiä	10
Laske arvosana	10
Laske taulukon alkioiden summa	10
Taulukon pienin arvo	10
Robotti	10
Pienin ja suurin arvo	10
Isot alkukirjaimet	10
Taulukon suurin frekvenssi	10
Käännä taulukko	10
Kokonaispistemäärä	90

Taulukko 2 Tenteistä kerätty data

5.1 Aineiston kerääminen, luokittelu ja analysointi

Aineistoa on kerätty kurssin aikana sekä kurssin päättökokeesta ViLLE -järjestelmän avulla. Kurssin aikana kerätty aineisto koostuu taulukon 1 mukaisesti. Kyseiset suoritteet on tehty suoraan ViLLE-järjestelmässä ja kerätty data on tätä kautta saatu kerättyä automaattisesti ViLLEN taustalla. Aineistosta on karsittu pois ne osallistujat, jotka eivät suorittaneet lopputenttiä tai osallistuivat pelkästään lopputenttiin, jotta voidaan vertailla, kuinka kurssiaktiivisuus on vaikuttanut lopulliseen tenttitulokseen. Kurssin päättökokeen aineistosta on yhdistetty kahden kurssikokeen vastausmäärät N=167 ja tehtävistä saadut pisteet. Nämä on käsitelty yhdessä kurssien päätöskokeiden samankaltaisuudesta johtuen. Analyysissa on käytetty Pearsonin korrelaatiota ja verrattu löytyykö suoraa korrelaatiota kurssiaktiivisuuden sekä tentin lopputuloksen kanssa. Datasta on muodostettu kuvaajia, joita on käytetty tutkimuksen apuna. Kvantitatiivisen tutkimuksen lisäksi on suoritettu kvalitatiivista tutkimusta kurssipalautteiden perusteella.

Pearsonin korrelaatiota on käytetty, jotta nähdään, löytyykö lineaarista riippuvuutta kurssiaktiivisuuden ja kurssin lopputentin kanssa. Kaavioita on käytetty datan selkeyttämiseksi ja tulkinnan helpottamiseksi.

5.2 Tutkimusaineiston valinta

Tutkimusaineistoksi valikoitui Turun yliopiston ensimmäisen vuosikurssin Algoritmien ohjelmoinnin peruskurssi, jotta pystytään tutkimaan vaikuttaako luento-opetus ja kurssiaktiivisuus opiskelijoiden tenttisuoritukseen ohjelmoinnissa. Lisäksi samalla päästään tutkimaan itse tenttiä ja ohjelmoinnin kurssien tenttimistä, kuinka hyvin tämä on onnistunut ja mitä mieltä opiskelijat ovat tästä. Kyseinen kurssi valittiin tutkimusaineiston saatavuuden ja mahdollisten taustavaikuttajien vähentämiseksi.

5.3 Kohteen valinta tutkimukselle

Tutkitaan, vaikuttaako oppilaiden kurssiaktiivisuus kurssin lopputulokseen eli tentin pisteisiin, jotta nähdään, onko kurssin opetustapa ja opetustyyli onnistunut. Pyritään löytämään vastaus, onko Aop-kurssin opetus onnistunutta, ja onko kurssin aikana suoritetuilla tehtävillä suoraa korrelaatiota tenttimenestykseen. Tuloksista riippuen voidaan mahdollisesti päätellä, onko kurssilla tarvetta lopputentille vai ovatko kurssin aikana tehdyt suoritukset riittäviä. Lisäksi pyritään selvittämään, onko kurssilla suoritettu tentti opiskelijoiden mielestä onnistunut, löytyykö korrelaatiota tentin tehtävien palautusmäärien ja pisteiden välillä, ja kuinka sähköinen tenttiminen on sopinut kyseiselle kurssille, jotta voidaan selvittää onko loppukokeessa parannettavaa.

Tutkimuskysymys 1: Vaikuttaako kurssiaktiivisuus tentin lopputulokseen?

Tutkimuskysymys 2: Minkä tasoinen algoritmien ja ohjelmoinnin peruskurssin päättökoe on?

Tutkimuskysymys 3: Millainen suhtautuminen opiskelijoilla on ohjelmoinnin sähköiseen tenttimiseen?

6 Tulokset

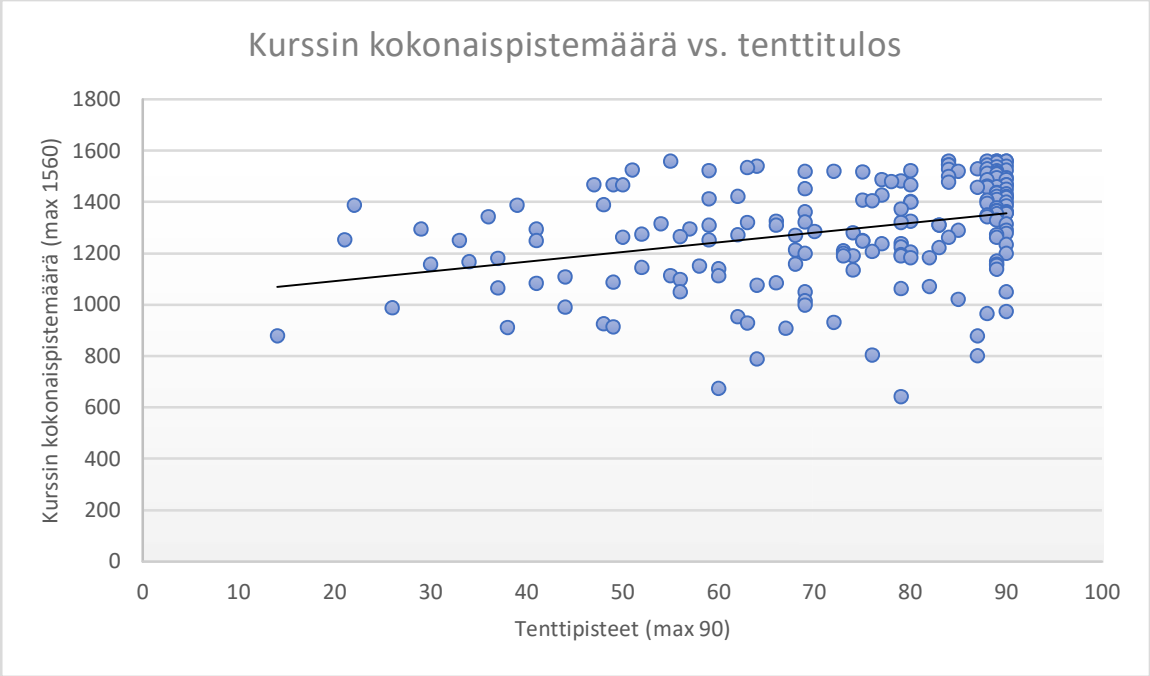
Tutkitaan kurssiaktiivisuutta Aop kurssin aikana kerätystä datasta ja selvitetään, löytyykö korrelaatiota tenttitulosten kanssa. Aktiivinen opetus ja muut opetustavat ovat osittain integroituna kyseiseen Aop-kurssiin. Demonstraatiot voidaan lukea aktiivisen oppimisen piiriin ja tutoriaalit parioppimisen sekä samalla aktiivisen oppimisen pariin. Motivaatio, oppilaiden kognitiiviset taidot, aiempi kokemus ohjelmoinnista, sukupuoli sekä ikä on rajattu tutkimuksen ulkopuolelle datan laadusta johtuen.

6.1 Kurssin rakenne

Aop koostuu viikoittaisista luennoista ja luentojen aiheisiin liittyvistä tutoriaaleista. Kurssin aikana suoritettiin demonstraatioita ja 4kpl harjoitustöitä. Kurssin läpäistäkseen harjoitustyöt oli suoritettava onnistuneesti ja demonstraatioista tuli saada vähintään puolet pisteistä. Pääosin kurssi suoritettiin ViLLE-järjestelmää käyttämällä. Kurssin päätteeksi suoritettiin tentti ViLLE -järjestelmällä valvotussa ympäristössä, joko omalla tietokoneella tai yliopiston mikroluokassa. Lisätietoja kurssista löytyy Turun yliopiston opinto-oppaasta (Turun yliopiston opinto-opas 5/2018).

6.2 Kurssiaktiivisuuden ja tenttitulosten analyysi valituilla menetelmillä

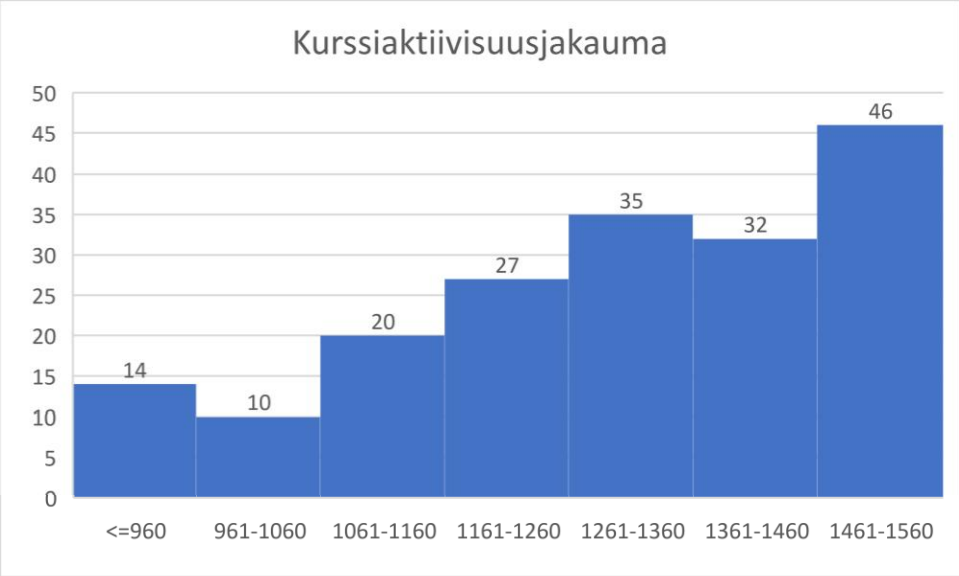
Kurssiaktiivisuutta mitattiin ViLLE -järjestelmästä saatujen tietojen perusteella ja näin mahdollisesti paikalla tapahtunut aktiivisuus on jätetty huomiotta tässä tutkimuksessa. Kuvassa 4 kurssin kokonaispistemäärää on verrattu tenttituloksiin.



Kuva 4 Kurssin kokonaispistemäärä määriteltynä kuvaajalle tenttitulosten kanssa

Kuvassa neljä nähdään kuinka N=184 opiskelijaa on jakautunut verrattaessa parhaita tenttipisteitä sekä kurssiaktiivisuutta. Kaaviosta nähdään, kuinka suurin osa opiskelijoista on sijoittunut kaavion oikeaan yläkulmaan. Muutoin opiskelijat ovat hajanaisesti sijoittuneet tenttipisteiden mukaan 20-80 välille ja 800 pisteen kurssiaktiivisuuden yläpuolelle. Korrelaatioksi tenttitulosten ja kurssin kokonaispistemäärän välille saatiin 0,34213. Korrelaatio demonstraatioiden sekä tentin lopputulosten osalta on 0,336.

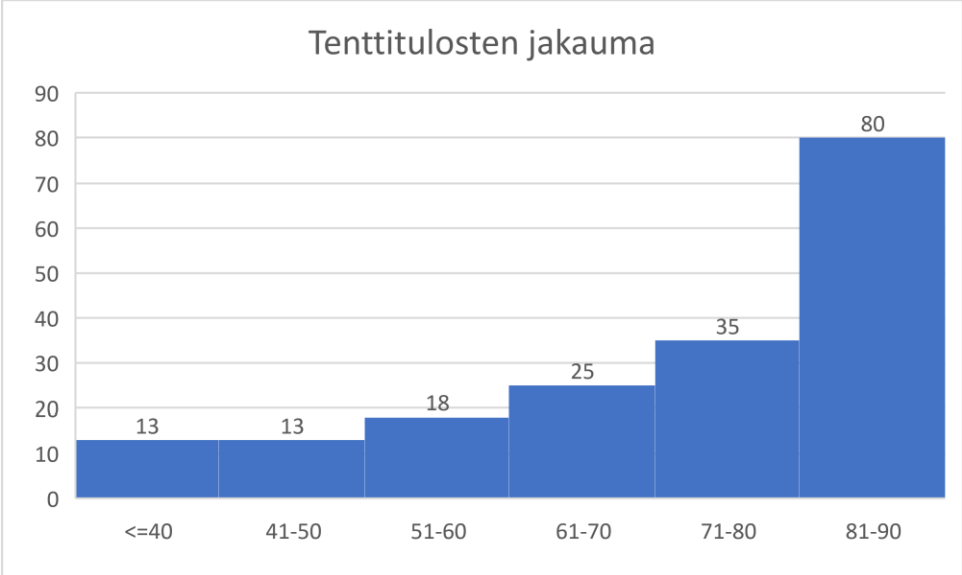
Kurssiaktiivisuuden pisteet ovat jakautununeet kuvan 5 mukaisesti.



Kuva 5 Kurssiaktiivisuus jaettuna lohkoihin

Kuvasta viisi nähdään, kuinka pääosa kurssiaktiivisuudesta on keskittynyt kuvion oikeaan reunaan yli 1261 pisteen.

Kuvassa 6 tenttitulokset jaettuna kaavioon.



Kuva 6 tenttitulosten jakauma

Kuvia viisi ja kuusi verrattaessa nähdään kuinka molemmat ovat kallellaan oikeaan reunaan. Noin 87% opiskeljoista on saavuttanut yli 1060 pistettä kurssin aikana ja 62,5% on saavuttanut yli 70 pistettä tentistä.

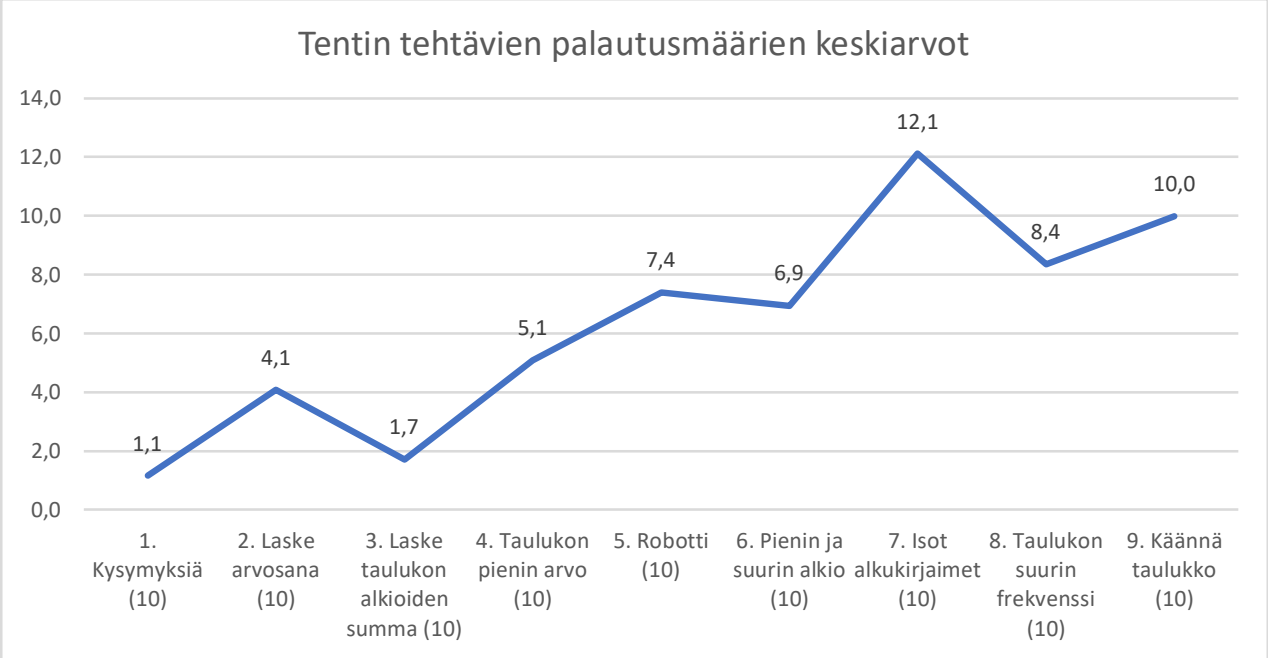
Taulukossa kolme kurssin aikana kerätyn datan keskiarvot verrattuina maksimipisteisiin.

Kurssin aikana kerätty data N=184	Keskiarvot	Maksimipisteet
Johdanto	19,8	20
7kpl ViLLE harjoituksia (1/vko)	296,7	27-52 (riippuen päivästä) yht. max. 322
Luentoläsnäolot	66,3	100
Tutoriaaliläsnäolot	6,3	7
Demonstraatiot 4krt	283,5	400
Pythonista Javaan	109	110
Muuttujat, merkkijonot ja ehtolauseet	94	95
Silmukat	97	100
Metodit	79	86
Taulukot	85	96
Valmiiden luokkien käyttö	80	100
Kertaus	60	75
Harjoitustyöt 4kpl	170,8	30 tai 60 yht. max.180
Kyselyt	49	50
Harjoitustentti	38,5	50
Tenttitulokset	72,9	90
Tenttipalaute	5	5
Kokonaispistemäärä/kurssiaktiivisuus	1291,31	1560

Taulukko 3 Kurssin aikana kerätyn datan keskiarvot

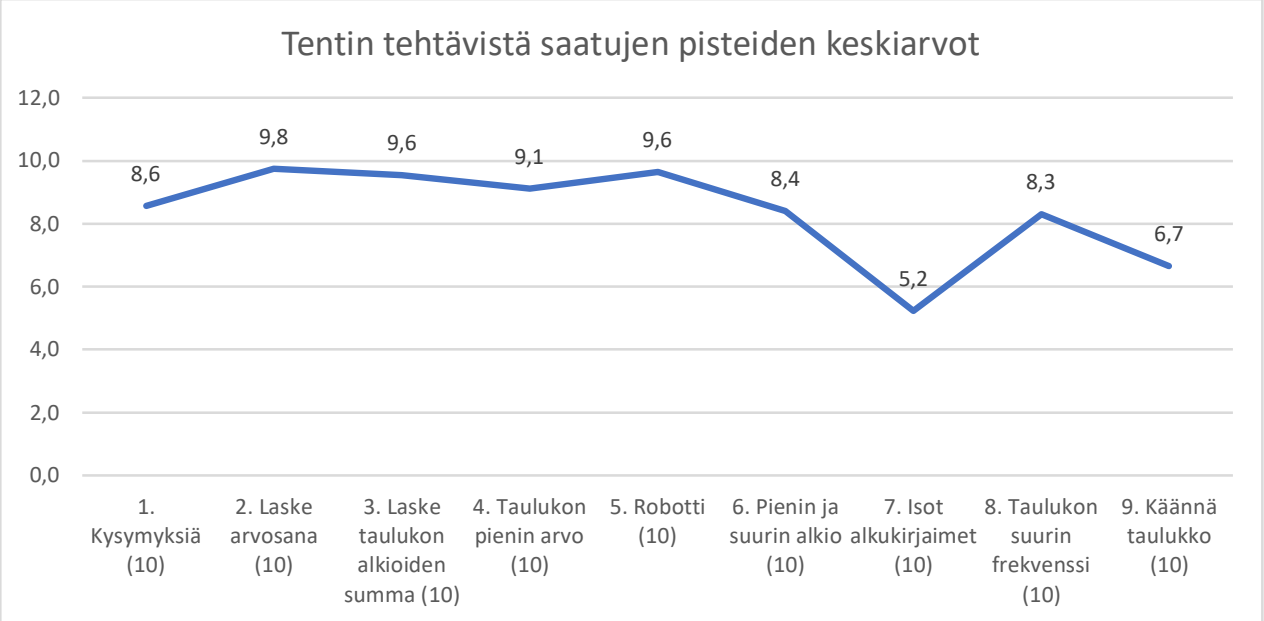
Mm. demonstraatioiden pisteet jäävät reilusti vajaaksi täysistä pisteistä, muiden aktiviteettien ollessa melko lähellä mahdollista maksimia. Demonstraatioiden pisteiden jääminen maksimista johtunee siitä, että muut aktiviteetit suoritetaan pareittain tai mahdollisesti luentojen aikana. Näin ainoaksi kotiin tehtäväksi asiaksi on jäänyt demonstraatiot. Harjoitustöiden korkeita pisteitä selittänee niiden pakollisuus. Mikäli oppilas ei suorittanut harjoitustöitä onnistuneesti ei tämä myöskään saavuttanut tenttioikeutta.

Kuvassa 7 kahden tentin palautusmäärien keskiarvot N=167. Datasta on poistettu palautusten määrän mukaan selkeästi poikkeavat palautusmäärät, jotka olivat yli 40. Kuvaajasta nähdään kuinka tehtävien numero 7 ja 9 palautusmäärät ovat huomattavasti korkeammat kuin muilla tehtävillä.



Kuva 7 Kahden tentin tehtävien palautusmäärien keskiarvot.

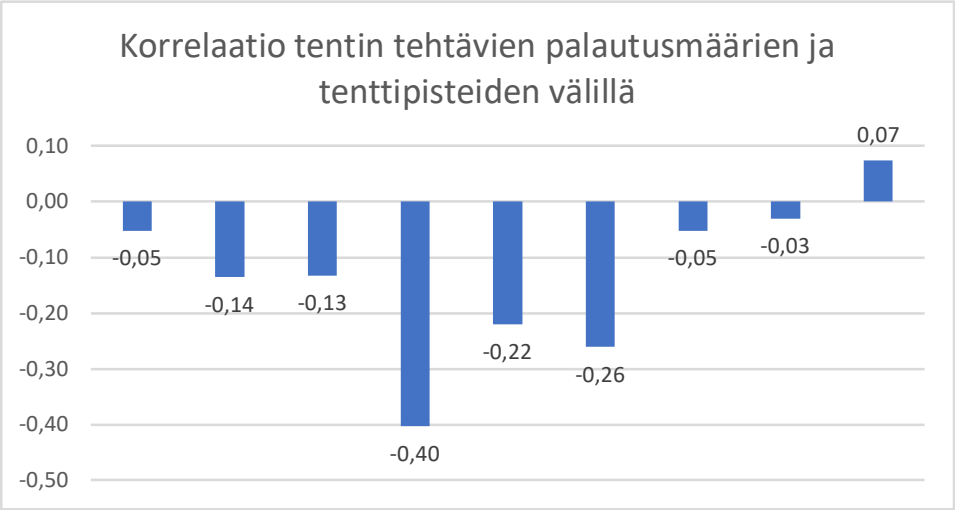
Kuvassa 8 tentin tehtävistä saatujen pisteiden keskiarvot. Kuvasta nähdään kuinka tehtävästä seitsemän ja yhdeksän on saatu selvästi vähemmän pisteitä kuin muista tehtävistä.



Kuva 8 Kahden tentin tehtävien pisteiden keskiarvot.

Samasta karsitusta datasta, josta edelliset kuvat on johdettu, selvitettiin korrelaatio tentin tehtävien palautusten ja saatujen tenttipisteiden välillä.

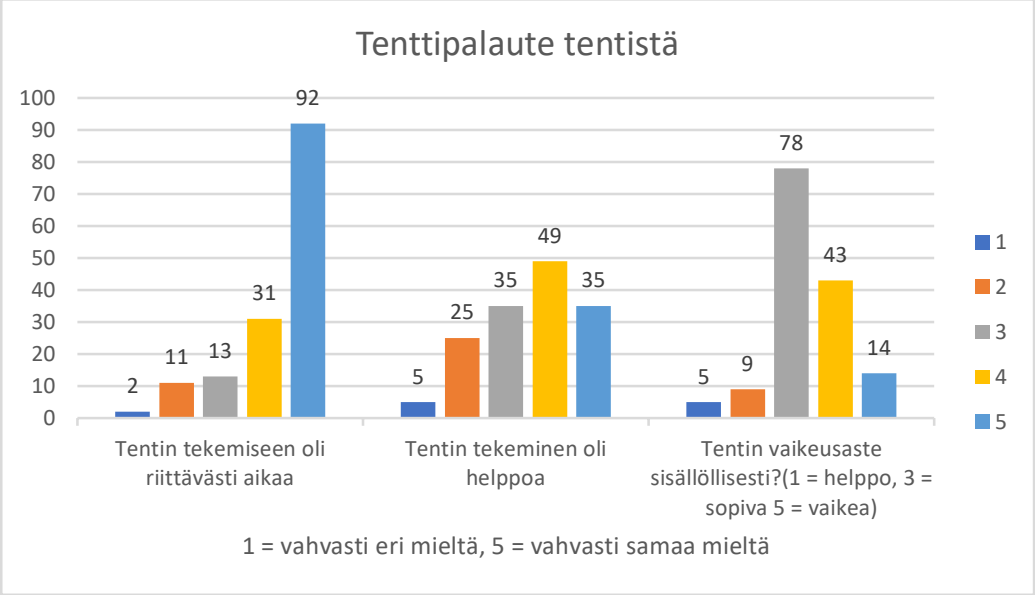
Kuvassa 9 merkitsevä negatiivinen korrelaatio (alle -0,2) löytyi vain tehtävien neljä-kuusi väliltä. Näissä tapauksissa suurimman korrelaation osoitti tehtävä neljä.



Kuva 9 korrelaatio tentin tehtävien palautusmäärien ja tenttipisteiden välillä

6.3. Tenttipalautteen analysointi valituilla menetelmillä

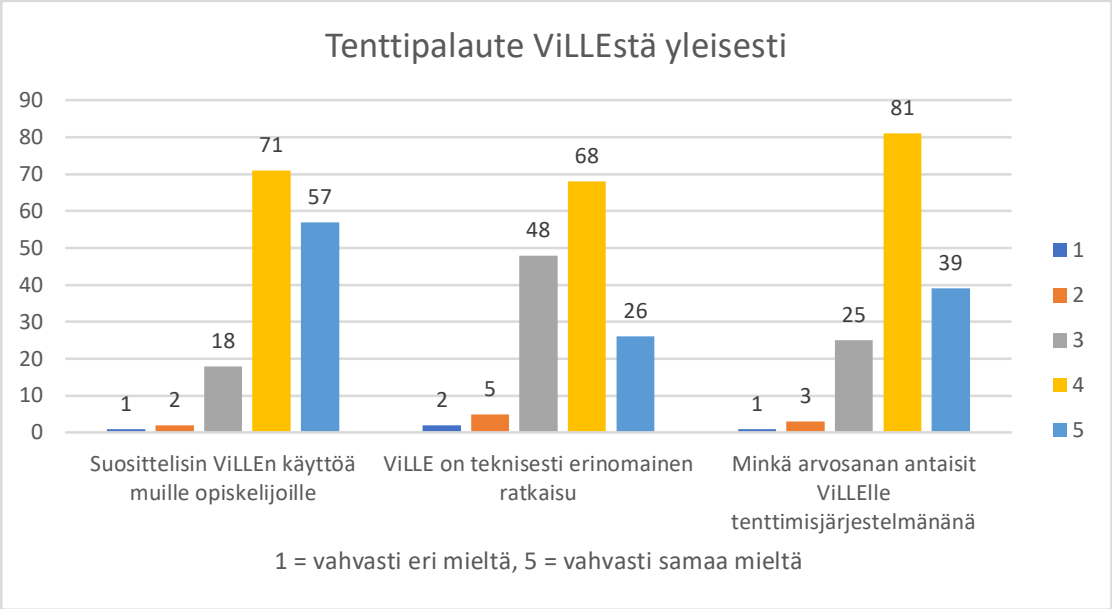
Tenttipalautetta antoi 149 opiskelijaa, jota on käytetty kuvissa 10-12. Kuvassa 10 tenttipalaute tentistä jaettuna vastausasteikoiden mukaan.



Kuva 10 tenttipalaute koskien tenttiä

Kuvasta 10 nähdään, kuinka suurin osa opiskelijoista oli sitä mieltä, että tentin tekemiseen oli riittävästi aikaa. Tentin tekeminen oli helppoa suurimman osan mielestä ja tentin vaikeusaste oli pääosan mielestä sopiva.

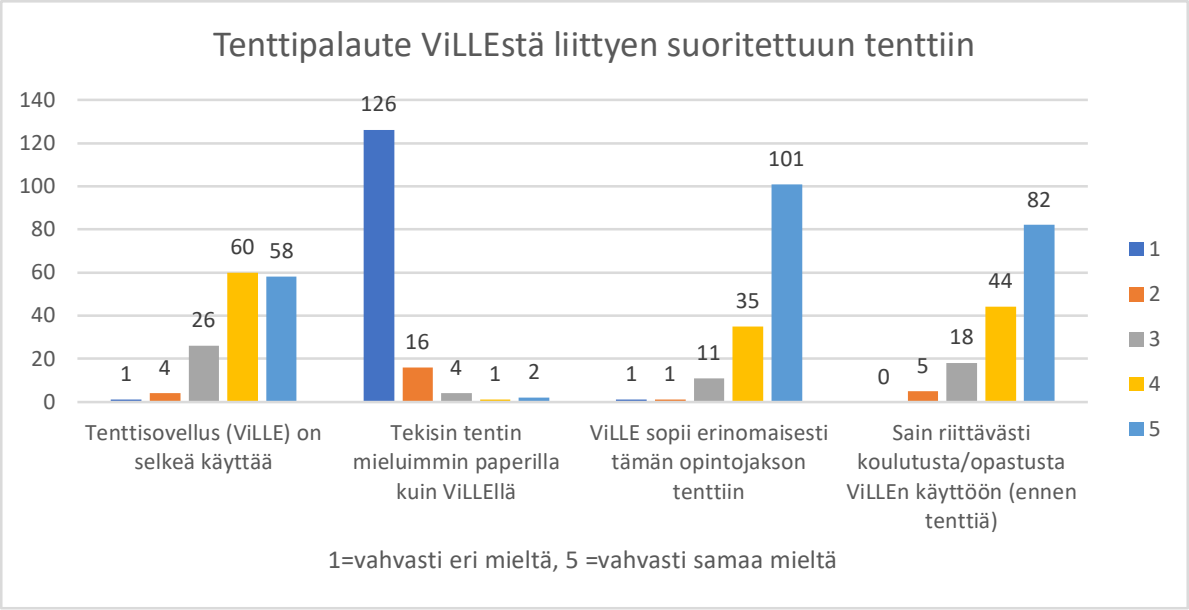
Kuvassa 11 tenttipalautetta ViLLE-oppimisjärjestelmästä jaettuna asteikolle 1-5.



Kuva 11 tenttipalaute käytetystä ViLLE järjestelmästä

Kuvasta 11 voidaan tulkita, että ViLLE -järjestelmää suositeltaisiin erittäin todennäköisesti muille opiskelijoille, ja ViLLE oli teknisesti toimiva ratkaisu. Arvosanana ViLLE -järjestelmä ansaitsisi numeron neljä. Kysyttäessä sujuiko tentti ilman teknisiä ongelmia, 128 henkeä vastasi kyllä ja 21 ei.

Kuvassa 12 tenttipalaute ViLLE -järjestelmästä liittyen suoritettuun tenttin asteikolla 1-5.



Kuva 12 tenttipalaute käytetystä ViLLE järjestelmästä liittyen suoritettuun tenttiin

Kuvasta 12 voidaan tulkita, kuinka ViLLEn käyttö oli suurimman osan mielestä selkeää sekä opiskelijat saivat riittävästi opastusta ViLLEn käyttöön ennen tenttiä. Valtaosa opiskelijoista on sitä mieltä, että he tekevät tentin mieluummin ViLLEssä kuin paperille. Lisäksi isoin osa opiskelijoista on sitä mieltä, että ViLLE sopii erinomaisesti Aop-kurssin tenttiin.

Taulukossa 4 kurssipalautteiden keskiarvot, asteikon ollessa 1=vahvasti eri mieltä ja 5= vahvasti samaa mieltä.

Tentistä kerätyn palautteen keskiarvot	Keskiarvot
Tentin tekemiseen oli riittävästi aikaa	4,3
Tentin tekeminen oli helppoa	3,6
Tenttisovellus (ViLLE) on selkeä käyttää	4,1
Tekisin tentin mieluimmin paperilla kuin ViLLEllä	1,2
ViLLE sopii erinomaisesti tämän opintojakson tenttiin	4,6
Tekisin tämän opintojakson tentin etätenttinä, jos se olisi mahdollista	3,9
Suosittelisin ViLLEn käyttöä muille opiskelijoille	4,2
ViLLE on teknisesti erinomainen ratkaisu	3,7
Minkä arvosanan antaisit ViLLElle tenttimisjärjestelmänä	4,0
Sain riittävästi koulutusta/opastusta ViLLEn käyttöön (ennen tenttiä)	4,4
Tentin vaikeusaste sisällöllisesti (1=helppo, 3=sopiva, 5=vaikea)	3,3

Taulukko 4 tenttipalautteiden keskiarvot

Taulukosta nähdään kuinka keskiarvot ovat pääosin neljän yläpuolella, lukuun ottamatta tentin tekemisen helppoutta, tentin tekoa ennemmin paperille, etätentin teko mahdollisuutta, ViLLEn teknistä ratkaisua ja tentin vaikeusastetta. Tentin vaikeusasteesta nähdään, kuinka keskiarvo on hyvin lähellä kolmea.

Avointa palautetta annettiin reilusti ja valtaosa tästä oli positiivista. Seuraavassa muutamia poimintoja oppilaiden kommenteista heidän vastatessaan mikä oli parasta ViLLEssä tenttimisessä.

”Järjestelmää on yksinkertaista käyttää”

”Sai palautetta koodaus tehtävien onnistumisesta.”

”Pääsee kunnolla ohjelmoimaan "oikeita" ohjelmia. Jos ohjelmointikurssien tentit kulminoituisivat ruutupaperille koodaamiseen, olisin varmasti aika nopeasti vaihtanut sivuainetta. Ville on pääosin erittäin hyvä oppimis- ja tenttiympäristö. Se on varsinkin hyvin stabiili oman kokemukseni mukaan, sillä se ei ole tainnut kaatua kertaakaan kahden kurssin aikana.”

”Koodaamistehtävia saa yrittää, kunnes onnistuu. Ei siis jää turhista virheistä kiinni. Jos koodi toimii, niin tuloksen näkee heti, eikä jää epävarmuutta onnistuiko tehtävä.”

”Se vastaa paperia paljon paremmin oikeaa koodausympäristöä”

”Parasta oli se, että oppimisympäristö oli tuttu. Kooditehtävissä parasta on se, että näkee oman vastauksensa suoritettuna (tietää menikö oikein).”

Positiivisissa palautteissa korostuu palautteen saaminen tehtävien onnistumisesta välittömästi koe tilanteessa sekä se että tentti vastaa lähemmin oikeaa koodausympäristöä kuin esimerkiksi kynä ja paperi.

Negatiivista palautetta annettiin hieman vähemmän ja nämä liittyivät pääosin ViLLEn teknisiin ongelmiin. Muutamia poimintoja vastauksista kysymykseen mitä parannuksia haluaisit ViLLEen / teknisiä ongelmia tentissä.

”Hieman selkeämpi ui”

”Suuret käyttäjämäärät samanaikaisesti tuntuvat vaikuttavan villeen. Esimerkiksi luennoilla ja tenteissä tulee välillä jäätymisiä, mikä johtaa javascript enginen kuolemiseen ja tilanteesta selviytyäkseen tätyy tehdä uusi sivulataus mikä vie aikaa tentin suorittamisesta.”

”Villen nopeudessa on parantelemisen varaa. Nousturi tehtävissä olisi kiva jos olisi mahdollista saada System.out.println() näkyviin jotta debugaus olisi paljon helpompaa”

”ViLLE jumittui tenttini aikana, mutta vika saatiin korjattua refreshillä (F5). ”

”Ville heitti ulos kolme kertaa tenttiä tehdessä ja oli erittäin hidas ajoittain. Tentin alusta kesti yli 10minuuttia, että pääsin tekemään itse tenttiä”

”Oli aika tylsä, että tentissä oli kaksi samantapaista tehtävää (etsi pienin ja etsi pienin ja suurin alkio). Kun ei saanut ekaa tehtyä, oli mahdotonta saada toistakaan toimimaan. Olisi kiva, että saisi käyttää vähän muitakin lisäapuja kuin API, koska oikeassa elämässä aina kuitenkin saatavilla lisätietoa muualta. ”

Vastauksissa toistuu selkeästi ViLLEn hidastuminen suuremmilla opiskelijamäärillä. Muita ongelmia on mainittu olevan, että opiskelijan ei ole sallittua käyttää APIn lisäksi muuta internettiä kuten normaalitilanteessa pystyisi.

Muutamia vastauksia kysymykseen mille muille opintojaksoille ViLLE sopisi hyvin saatiin seuraavanlaisia vastauksia.

”Lähes kaikilla kursseilla olisi hyvä olla ViLLE käytössä lukuunottamatta esim. matematiikan tenttejä, joissa on helpompi ratkaista tehtävä paperilla”

”Kaikelle ohjelmoinnille. Sopisi myös mahdollisesti kauppakorkean laskentatoimen kursseille.”

”Biologian pääaineopiskelijana näkisin sovellutuksia ViLLEn käytössä esimerkiksi näytteiden tunnistuksia harjoitellessa. Tällä hetkellä kurssitehtävät löytyvät Moodlesta, mutta se on joidenkin tehtävien, etenkin laskutehtävien yhteydessä kömpelöhköä.”

”Teoriassa mihin vaan. Käytännössä parhaiten se sopii kursseille, joissa tietokoneen käyttö on paljon läsnä.”

”Mielestäni pelillistäminen ja opetusohjelmat sopivat lähes mihin vaan opetukseen sovellettuna.”

”Lähinnä ohjelmointikursseille. Fysiikan opiskelijana olen sitä mieltä, että fysiikan ja matematiikan tenttejä varten matemaattisten kaavojen syöttöön pitäisi olla lisäksi piirtopöydältä

Tex (ja siitä PDF)-muotoon muuttava kuvantunnistusohjelma, sillä näillä aloilla koko ratkaisun arviointi välivaiheineen välttämätöntä osaamisen testaamista varten. Toistaiseksi olen nähnyt matematiikalla ja fysiikalla vain paperitenttejä."

"Mielestäni villetehtävät sopii kaikille opintojaksoille. Se, että voi tehdä harjoitustehtäviä luentojen lisäksi villen kaltaisessa ympäristössä on hyödyksi."

Vastauksista löydetään yhteiseksi tekijäksi tietokoneen käyttö. ViLLEä käytettäisiin mieluusti muilla kursseilla, joissa muutenkin käytetään tietokonetta. Epäilystä aiheuttivat mm. matematiikan ja fysiikan kaavat, kuinka ne sopivat ViLLE alustalle.

7 Johtopäätökset

Kurssiaktiivisuuden ja tenttitulosten väliltä löytynyt korrelaatio 0,34 ilmentää heikkoa riippuvuutta näiden kahden muuttujan välillä. Kuten taulukosta 1 selviää suurin osa opiskelijoista, on suorittanut yli 1200 pistettä kurssin aikana ja saavuttanut tentistä yli 80 pistettä. Näistä voidaan arvella, että löytyvä korrelaatio olisi suurempi kuin muista oppilaista, joiden tulokset hajaantuvat taulukossa alle 80 pisteen. Syitä sille mistä korrelaation odotettua pienempi suuruus johtuu, voitaneen lukea johtuneen tutkimuksessa poisjätetyistä määreistä. Datan laadusta johtuen poisjätetyt määreet olivat motivaatio, sukupuoli, ikä, kognitiiviset taidot sekä aiempi kokemus ohjelmoinnista. Muita mahdollisia vaikuttajia ovat saattaneet olla kurssilla käytetyt opetustyylit. Mm. tutoriaalien aikana käytetty parioppiminen on saattanut mahdollistaa osalle opiskelijoista ns. vapaamatkustaja vaikutuksen. Tällöin toinen opiskelijoista on saattanut saada hyviä pisteitä kurssin aikana osallistumatta joko hyvin vähän, mikäli ollenkaan tehtävien tekoon. 62,5% opiskelijoista sai tentistä kuitenkin yli 70 pistettä. Täten voidaan todeta, että tentti ei ollut liian vaikea kurssin opiskelijoille. Tätä väitettä tukee tenttipalaute, jonka mukaan suurin osa opiskelijoista oli sitä mieltä, että tentin vaikeusaste oli sopiva. Näin voitaneen todeta, että kurssiaktiivisuudella on osittainen vaikutus tentin lopputulokseen, mutta tämä ei ole ainoa osatekijä kurssin lopputuloksen kannalta.

Muita mahdollisia korrelaatioita pyrittiin etsimään annetusta datasta. Demonstraatioiden ja tenttitulosten korrelaatioksi osoittautui 0,336 joka on hyvin lähellä koko kurssiaktiivisuuden ja tenttitulosten välille löytynyttä korrelaatiota. Tämä todistanee, että demonstraatioita tekemällä pystyy vaikuttamaan positiivisesti kurssin lopputulokseen, vaikka korrelaatio on heikko. Tätäkään ei siis voida pitää ainoana osatekijänä hyvän lopputuloksen kannalta.

Korrelaatio luentoläsnäolojen ja tenttipisteiden kanssa oli 0,02 ja tutoriaalien läsnäolojen korrelaatio tenttipisteiden kanssa oli 0,09. Kummastakaan näistä ei löydetty korrelaatiota mutta tämä ei tarkoita sitä, että kyseiset osat olisivat turhia tai merkityksettömiä kurssin kokonaisuuden kannalta.

Tenttitehtävien pisteissä ja tenttitehtävien palautusten määrässä nähdään heikko merkitsevä korrelaatio tehtävissä 4-6. Tämä johtunee siitä, että kyseiset tehtävät osuivat vaikeusasteeltaan juuri sopivaan kohtaan. Ajankäytöllä on saattanut olla myös vaikutusta tuloksiin, sillä mikäli opiskelija on kokenut tehtävät 4-6 juuri sopivan vaikeiksi on näiden ratkaisuun saattanut mennä paljon aikaa. Tästä syystä lopuille tehtäville jäänyt aika on saattanut verottaa keskittymistä loppuihin, vielä hieman vaikeampiin tehtäviin. Osalta opiskelijoista onkin jäänyt tehtävät 7-9 kokonaan tekemättä, eivätkä he ole edes yrittäneet viimeisiä 2-3 tehtävää. Ensimmäiset kolme

tehtävää ovat puolestaan olleet laadultaan selvästi helpompia kuin muut tehtävät keskimääräisten palautusmäärien ollessa pienet ja keskimääräiset pisteet kuitenkin korkeat. Tämä osaltaan johtuu tehtävien laadusta mm. ensimmäisen tehtävän ollessa monivalinta, jolloin opiskelija ei ole saanut palautetta vastauksistaan ja on näin siirtynyt eteenpäin. Tehtävä numero 7 on osoittautunut haastavimmaksi tehtäväksi keskimääräisen palautusmäärien kohotessa peräti 12 ja keskimääräisten pisteiden jäädessä vain 5,2. Palautusmäärien ja saatujen pisteiden välille löytyi heikkoa korrelaatiota tehtävien 4 ja 6 kohdalla. Näistä etenkin tehtävän 4 kohdalla nähdään hieman selkeämpi -0.40 korrelaatio kuin 6 tehtävässä oleva -0.26. Tehtävän neljä suurempi korrelaatio johtunee tehtävän tyypistä ja helppoudesta. Vastausmäärät olivat kaikilla opiskelijoilla yhtä lukuun ottamatta 5 tai alle. Kaikki opiskelijat saivat tehtävästä myös pisteitä joko 5 tai 10 kahta opiskelijaa lukuun ottamatta, jotka saivat tehtävästä 0 pistettä. Tehtävästä oli mahdollista saada 0, 5 tai 10 pistettä ja koska tehtävä oli selkeästi helpomman puoleinen, löydettiin tästä korrelaatiota. Tehtävän numero viisi korrelaatiota selittänee tehtävän samankaltaisuus luennoilla käytyjen harjoitteiden kanssa.

Tentistä saadut pisteet sekä tentin pisteiden keskiarvo 72,9 osoittavat, että tutkimuskysymys 2 ”minkä tasoinen aopin tentti on” voidaan vastauksen todeta olevan, että tentti on helpomman puoleisen ja sopivan välimaastossa. Tätä tukemaan on tenttipalaute, jossa opiskelijoista 78 on sitä mieltä, että tentin vaikeusaste on sopiva sekä kuvan 6 kaavion jakauma, joka on reilusti kallellaan oikeaan reunaan.

Tenttipalautteen perusteella ViLLE itsessään oli onnistunut tenttialusta ja opiskelijat suosivat sähköistä tenttiä etenkin tässä tapauksessa. Perusteluina, palautteiden perusteella, oli tentin sijoittuminen ohjelmoinnin oikeaan ympäristöön paperisen tentin sijaan. Ongelmakohtia oli enimmäkseen ViLLEn rasituksen kestävyydessä sekä muutamissa muissa teknisissä ongelmissa. Opiskelijat suosittelisivat ViLLEä erittäin mielellään muille opiskelijoille. ViLLElle annettiin tenttimisalustana myös arvosanaksi pääosin 4-5, joten ViLLEä tenttimisalustana voidaan pitää erittäin onnistuneena. Näin pystytään poissulkemaan mahdollisen tenttimisalustan vaikutus sähköiseen tenttimiseen. Palaute itse tentistä osoittaa, että tentin tekemiseen oli riittävästi aikaa, tentin vaikeusaste oli sopiva ja tentin tekeminen oli helppoa. Näistä sekä avoimista palautteista voidaan todeta, että vastaus tutkimuskysymykseen ”miten opiskelijat suhtautuvat ohjelmoinnin sähköiseen tenttimiseen” voidaan todeta olevan positiivisesti sekä opiskelijat suosivat suuresti sähköistä tenttiä paperisen tentin sijaan.

Mikäli ViLLEä käytettäisiin muissa opintojaksoissa olisi tämä opiskelijoiden mielestä sovellettuna mahdollisesti toimiva ratkaisu. Suoraan samanlaisella toteutuksella, kuin Aop -kurssilla oli käytetty ViLLEä, nähtiin käytettävän pääosin vain muilla ohjelmointikursseilla. Tästä

voidaan päätellä, että ViLLE miellytti opiskelijoita sen takia, että se sopi käyttötarkoitukseensa niin hyvin. Oppimisjärjestelmän käyttöönottoa muilla kursseilla ei tyrmätty, mutta käyttöä tulisi soveltaa tietyin reunaehdoin, jotta käyttö olisi miellyttävää.

8 Yhteenveto

Tutkimuksen mukaan löydettiin lievää korrelaatiota kurssiaktiivisuuden ja tenttitulosten välillä. Odotettua heikompi korrelaatio johtunee kurssin opetustyylin mahdollistavasta vapaamatkustamisesta sekä huomiotta jätetyistä määreistä, kuten aikaisempi kokemus ohjelmoinnista sekä kognitiivisista taidoista. Turun yliopiston ensimmäisen ohjelmointikurssi Aopin tentin voidaan todeta olevan onnistunut kokonaisuus vaikeusasteeltaan. Ohjelmointi itsessään tapahtuu sähköisessä muodossa, joten kuten odotettua, ohjelmoinnin tentti tehdään ennemmin sähköisenä kuin paperille. Kyseiset tutkimustulokset olivat melko odotettuja. Ainoaksi pohdinnan aiheeksi jäi kurssiaktiivisuuden ja tenttitulosten heikko korrelaatio. Tätä olisi syytä tutkia tulevaisuudessa uusintatutkimuksella, jotta voitaisiin selvittää mitkä muut tekijät vaikuttavat tenttituloksiin. Näin voitaisiin myös verifioida tästä tutkimuksesta löytyvät tulokset.

Muita mahdollisia selittäjiä odotettua heikommalle korrelaatiolle saattaa olla kurssin aikana suoritettujen tehtävien helppous verrattaessa tenttiin. Opiskelija on pystynyt yrittämään kurssin tehtäviä niin paljon kuin mahdollista kurssin aikana ja näin mahdollisesti pystynyt keräämään täydet pisteet osittain arvaamalla. Käytössä on saattanut olla yritys ja erehdys metodiikka, jolloin opiskelija on yrittänyt suorittaa tehtävää yrittämällä useaan otteeseen ja korjannut vastaustaan yritysten perusteella. Sen sijaan, että opiskelija olisi ymmärtänyt ongelman itsessään ja vastannut kysymykseen syvemmän ymmärryksen kautta.

Kokonaisvaltaisen oppimisjärjestelmän käyttö kurssin aikana on helpottanut opiskelijoiden opiskelua. Riittävästi saatu tuki järjestelmän käytössä on poistanut mahdollisen järjestelmän käytön ahdistuksen. Osittain ehkä juuri oppimisjärjestelmän takia, kurssin oppimistulokset ovat hyvät verrattaessa vastaavaan kurssiin ennen ViLLEn käyttöönottoa. ViLLEä suositeltiin käytettäväksi muilla kursseilla, mahdollisesti tarpeen vaatiessa soveltaen. Tämä osaltaan kertoo, kuinka järjestelmä on onnistunut ja kuinka tyytyväisiä opiskelijat ovat ViLLEn käyttöön. Mikäli tentissä olisi käytetty erillistä tenttimisalustaa, olisi tämä saattanut osaltaan aiheuttaa ahdistusta opiskelijoiden keskuudessa ja täten mahdollisesti huonompia lopputuloksia.

8.1 Tulevaisuudessa

Tulevaisuudessa olisi mielenkiintoista suorittaa laajempi tutkimus, jossa pystytään ottamaan huomioon lisämääreitä kuten motivaatio ja muut mahdolliset taustavaikuttajat. Näin saataisiin mahdollisesti eliminoitua muuttujia ja löydettäisiin syvempiä korrelaatioita, jotka vaikuttavat tenttituloksiin. Olisi myös suotavaa pyrkiä huomioimaan mahdollinen vapaamatkustaja vaikutus ja selvittää kuinka paljon tätä esiintyy kyseisellä kurssilla. Lisäksi olisi syytä tutkia miksi kurssilla

aktiivisesti osallistuneet opiskelijat eivät suorita tenttiä, ja tämän myötä kurssia tarvittaessa muokata parempaan suuntaan. Mielenkiintoista olisi myös tutkia, kuinka suurempi kurssin pelillistäminen tai muut opetusmetodit vaikuttaisivat kurssin lopputuloksiin, ja kuinka opiskelijat vastaanottaisivat kyseiset opetusmetodit.

Lähteet

Aarabi, P., Norouzi, N., Wu, J. & Spears, M. 2016, "7 surprising lessons learned from teaching iOS programming to 30,000+ MOOC students", , pp. 1.

Ahmadzadeh, M., Elliman, D. & Higgins, C. 2005, "An Analysis of Patterns of Debugging Among Novice Computer Science Students", *SIGCSE Bull.,* vol. 37, no. 3, pp. 84-88.

Al-Tahat, K., Taha, N., Hasan, B. & Shawar, B.A. 2016, "The impact of a 3D visual tool on female students attitude and performance in computer programming", , pp. 864.

Andreatos, A.S. 2009, "Electronic exams for the 21st century", *Proceedings of the European Computing Conference*Springer, , pp. 383.

Bennedsen, J. & Caspersen, M.E. 2007, "Failure Rates in Introductory Programming", *SIGCSE Bull.,* vol. 39, no. 2, pp. 32-36.

Bergin, S. & Reilly, R. 2005, "Programming: Factors That Influence Success", *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*ACM, New York, NY, USA, pp. 411.

Brink, R. & Lautenbach, G. 2011, "Electronic assessment in higher education", *Educational Studies,* vol. 37, no. 5, pp. 503-512.

C. L. Gunn & J. Raven 2017, "Smart education: Introducing active learning engineering classrooms in the Middle East", *2017 Fourth HCT Information Technology Trends (ITT)*, pp. 1.

Crisp, G. 2011, "Teacher's Handbook on e-Assessment", *Transforming Assessment-An ALTC Fellowship Activity,* vol. 18.

Davies, W.M. 2009, "Groupwork as a form of assessment: Common problems and recommended solutions", *Higher Education,* vol. 58, no. 4, pp. 563-584.

Dierbach, C., Taylor, B., Zhou, H. & Zimand, I. 2005, "Experiences with a CS0 Course Targeted for CS1 Success", *SIGCSE Bull.,* vol. 37, no. 1, pp. 317-320.

Dykman, C.A. & Davis, C.K. 2008, "Online education forum: Part two-teaching online versus teaching conventionally", *Journal of Information Systems Education,* vol. 19, no. 2, pp. 157-164.

Engle, S. & Rollins, S. 2013, "Expert Code Review and Mastery Learning in a Software Development Course", *J.Comput.Sci.Coll.,* vol. 28, no. 4, pp. 139-147.

Falchikov, N. & Goldfinch, J. 2000, "Student Peer Assessment in Higher Education: A Meta-Analysis Comparing Peer and Teacher Marks", *Review of Educational Research,* vol. 70, no. 3, pp. 287-322.

Freeman, J., Magerko, B., McKlin, T., Reilly, M., Permar, J., Summers, C. & Fruchter, E. 2014, "Engaging Underrepresented Groups in High School Introductory Computing Through

Computational Remixing with EarSketch", *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*ACM, New York, NY, USA, pp. 85.

Freeman, S., Eddy, S.L., McDonough, M., Smith, M.K., Okoroafor, N., Jordt, H. & Wenderoth, M.P. 2014, "Active learning increases student performance in science, engineering, and mathematics", *Proceedings of the National Academy of Sciences,* vol. 111, no. 23, pp. 8410-8415.

G. Sukadarmika, R. S. Hartati, Linawati, N. P. Sastra, D. M. Wiharta & M. A. Setiawan 2016, "Proposed model for e-exam availability in WLAN environment", *2016 International Conference on Smart Green Technology in Electrical and Information Systems (ICSGTEIS)*, pp. 89.

Giangrande, J.,Ernie 2007, "CS1 Programming Language Options", *J.Comput.Sci.Coll.,* vol. 22, no. 3, pp. 153-160.

Graham, C., Cagiltay, K., Lim, B., Craner, J. & Duffy, T.M. 2001, "Seven principles of effective teaching: A practical lens for evaluating online courses", *The technology source,* vol. 30, no. 5, pp. 50.

Guskey, T.R. 2007, "Closing Achievement Gaps: Revisiting Benjamin S. Bloom's â€œLearning for Masteryâ€• ", *Journal of Advanced Academics,* vol. 19, no. 1, pp. 8-31.

H. A. Bazar 2017, "Forms distribution algorithm for online examination systems", *2017 8th International Conference on Information Technology (ICIT)*, pp. 123.

I. Simonova & P. Poulova 2016, "Assessment preferences of IT and management students", *2016 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*, pp. 58.

Iosup, A. & Epema, D. 2014, "An Experience Report on Using Gamification in Technical Higher Education", *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*ACM, New York, NY, USA, pp. 27.

J. Dreier, R. Giustolisi, A. Kassem, P. Lafourcade, G. Lenzini & P. Y. A. Ryan 2014, "Formal analysis of electronic exams", *2014 11th International Conference on Security and Cryptography (SECRYPT)*, pp. 1.

J. Elmaleh & V. Shankararaman 2017, "Improving student learning in an introductory programming course using flipped classroom and competency framework", *2017 IEEE Global Engineering Education Conference (EDUCON)*, pp. 49.

J. M. Bekki, O. Dalrymple & C. S. Butler 2012, "A mastery-based learning approach for undergraduate engineering programs", *2012 Frontiers in Education Conference Proceedings*, pp. 1.

Jenkins, T. 2001, "The Motivation of Students of Programming", *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*ACM, New York, NY, USA, pp. 53.

Kaila, E., Rajala, T., Laakso, M., Lindén, R., Kurvinen, E., Karavirta, V. & Salakoski, T. 2015, "Comparing student performance between traditional and technologically enhanced programming course.", *ACE,* vol. 160, pp. 147-154.

Kaila, E., Rajala, T., Laakso, M., Lindén, R., Kurvinen, E. & Salakoski, T. 2014, "Utilizing an Exercise-Based Learning Tool Effectively in Computer Science Courses.", *Olympiads in Informatics,* vol. 8.

Kaila, E., Rajala, T., Laakso, M.-. & Salakoski, T. 2009, "Effects, Experiences and Feedback from Studies of a Program Visualization Tool", *Informatics in Education,* vol. 8, pp. 17{\tetendash}34.

King, P.E. & Behnke, R.R. 2005, "Problems Associated With Evaluating Student Performance In Groups", *College Teaching,* vol. 53, no. 2, pp. 57-61.

Kuikka, M., Kitola, M. & Laakso, M. 2014, "Challenges when introducing electronic exam", *Research in Learning Technology,* vol. 22, no. 0.

Kulik, C.C., Kulik, J.A. & Bangert-Drowns, R.L. 1990, "Effectiveness of Mastery Learning Programs: A Meta-Analysis", *Review of Educational Research,* vol. 60, no. 2, pp. 265-299.

Kurvinen, E., Hellgren, N., Kaila, E., Laakso, M. & Salakoski, T. 2016, "Programming Misconceptions in an Introductory Level Programming Course Exam", *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*ACM, New York, NY, USA, pp. 308.

L. Junqiang, Z. Jing & L. Jianghua 2017, "Practice and study of the flipped classroom based on active learning ability enrichment", *2017 12th International Conference on Computer Science and Education (ICCSE)*, pp. 591.

L. Moccozet, C. Tardy, W. Opprecht & M. Léonard 2013, "Gamification-based assessment of group work", *2013 International Conference on Interactive Collaborative Learning (ICL)*, pp. 171.

Laakso, M., Rajala, T., Kaila, E. & Salakoski, T. 2008, "The impact of prior experience in using a visualization tool on learning to program", *Appeared in Cognition and Exploratory Learning in Digital Age (CELDA 2008),* , pp. 13-15.

Lahtinen, E., Ala-Mutka, K. & J\arvinen, H. 2005, "A Study of the Difficulties of Novice Programmers", *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*ACM, New York, NY, USA, pp. 14.

Law, K.M.Y., Lee, V.C.S. & Yu, Y.T. 2010, *Learning motivation in e-learning facilitated computer programming courses*.

Leithner, A. 2011, "Do Student Learning Styles Translate to Different â€œTesting Stylesâ€• ?", *Journal of Political Science Education,* vol. 7, no. 4, pp. 416-433.

Lishinski, A., Yadav, A., Good, J. & Enbody, R. 2016, "Learning to Program: Gender Differences and Interactive Effects of Students' Motivation, Goals, and Self-Efficacy on

Performance", *Proceedings of the 2016 ACM Conference on International Computing Education Research*ACM, New York, NY, USA, pp. 211.

Lokkila, E., Kaila, E., Lindén, R., Laakso, M. & Sutinen, E. 2017, "Refactoring a CS0 course for engineering students to use active learning", *Interactive Technology and Smart Education,* vol. 14, no. 3, pp. 182-195.

M. A. Trpkovska, L. A. Bexheti & B. Cico 2017, "Enhancing flipped classroom model implementation", *2017 6th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1.

M. Dorodchi, N. Dehbozorgi & T. K. Frevert 2017, ""I wish I could rank my exam's challenge level!": An algorithm of Bloom's taxonomy in teaching CS1", *2017 IEEE Frontiers in Education Conference (FIE)*, pp. 1.

M. Hayat, R. Hasan, S. I. Ali & M. Kaleem 2017, "Active learning and student engagement using Activity Based Learning", *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, pp. 201.

M. Kohana & S. Okamoto 2016, "A Reservation System for Oral Exam of Programming Class", *2016 19th International Conference on Network-Based Information Systems (NBiS)*, pp. 516.

M. O. Smith, A. Giugliano & A. DeOrio 2017, *Long Term Effects of Pair Programming*.

M. S. Patil, P. Desai, M. Vijayalakshmi, M. M. Raikar, S. Battur, H. Parikshit & G. H. Joshi 2016, "Trusted Relative Peer Review: A Novel Approach to Assess an Individual in Team Based Learning", *2016 IEEE 4th International Conference on MOOCs, Innovation and Technology in Education (MITE)*, pp. 54.

M. Seyam, D. S. McCrickard, S. Niu, A. Esakia & W. Kim 2016, "Teaching mobile application development through lectures, interactive tutorials, and Pair Programming", *2016 IEEE Frontiers in Education Conference (FIE)*, pp. 1.

Malliarakis, C., Satratzemi, M. & Xinogalos, S. 2017, "CMX: The Effects of an Educational MMORPG on Learning and Teaching Computer Programming", *IEEE Transactions on Learning Technologies,* vol. 10, no. 2, pp. 219-235.

McCane, B., Ott, C., Meek, N. & Robins, A. 2017, "Mastery Learning in Introductory Programming", *Proceedings of the Nineteenth Australasian Computing Education Conference*ACM, New York, NY, USA, pp. 1.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. 2001, "A Multi-national, Multi-institutional Study of Assessment of Programming Skills of First-year CS Students", *SIGCSE Bull.,* vol. 33, no. 4, pp. 125-180.

N. Salleh, E. Mendes & J. Grundy 2011, "Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review", *IEEE Transactions on Software Engineering,* vol. 37, no. 4, pp. 509-525.

Naps, T.L., R\o\ssling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. & Vel\'azquez-Iturbide, J.\. 2002, "Exploring the Role of Visualization and Engagement in Computer Science Education", *SIGCSE Bull.,* vol. 35, no. 2, pp. 131-152.

O. S. Yan & G. Cheng 2017, "Exploring the impact of flipped classroom on students' acceptance of programming in secondary education", *2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pp. 246.

P. Dębiec 2018, "Effective Learner-Centered Approach for Teaching an Introductory Digital Systems Course", *IEEE Transactions on Education,* vol. 61, no. 1, pp. 38-45.

P. E. Robinson & J. Carroll 2017, "An online learning platform for teaching, learning, and assessment of programming", *2017 IEEE Global Engineering Education Conference (EDUCON)*, pp. 547.

P. Seeling & J. Eickholt 2017, "Levels of active learning in programming skill acquisition: From lecture to active learning rooms", *2017 IEEE Frontiers in Education Conference (FIE)*, pp. 1.

Partanen, T., Mannila, L. & Poranen, T. 2016, "Learning Programming Online: A Racket-course for Elementary School Teachers in Finland", *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*ACM, New York, NY, USA, pp. 178.

R. Giustolisi, G. Lenzini & G. Bella 2013, "What security for electronic exams?", *2013 International Conference on Risks and Security of Internet and Systems (CRiSIS)*, pp. 1.

R. Hijón-Neira, L. Santacruz-Valencia, D. Pérez-Marín & M. Gómez-Gómez 2017, "An analysis of the current situation of teaching programming in Primary Education", *2017 International Symposium on Computers in Education (SIIE)*, pp. 1.

R. T. Selvi & G. Chandramohan 2016, "Peer Assessment of Oral Presentation: An Investigative Study of Using Clickers in First-Year Civil Engineering Class of a Reputed Engineering Institution", *2016 IEEE Eighth International Conference on Technology for Education (T4E)*, pp. 132.

Rajala, T., Kaila, E., Lind\'en, R., Kurvinen, E., Lokkila, E., Laakso, M. & Salakoski, T. 2016, "Automatically Assessed Electronic Exams in Programming Courses", *Proceedings of the Australasian Computer Science Week Multiconference*ACM, New York, NY, USA, pp. 11:1.

Robins, A., Rountree, J. & Rountree, N. 2003, "Learning and teaching programming: A review and discussion", *Computer science education,* vol. 13, no. 2, pp. 137-172.

Rubio, M.A., Romero-Zaliz, R., Mañoso, C. & de Madrid, A.P. 2015, *Closing the gender gap in an introductory programming course*.

Ruegg, W. 2004, "A History of the University in Europe", *Volume,* vol. 3, pp. 1800-1945.

S. Alhazbi 2016, "Using flipped classroom approach to teach computer programming", *2016 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pp. 441.

S. España, D. Insa, J. Silva & S. Tamarit 2017, "In what order should i correct the exercises? Determining the evaluation order for the automatic assessment of programming exercises", *2017 16th International Conference on Information Technology Based Higher Education and Training (ITHET)*, pp. 1.

Sabitzer, B. & Pasterk, S. 2014, "Brain-based programming continued: Effective teaching in programming courses", , pp. 1.

Settle, A., Vihavainen, A. & Sorva, J. 2014, "Three Views on Motivation and Programming", *Proceedings of the 2014 Conference on Innovation \&\#38; Technology in Computer Science Education*ACM, New York, NY, USA, pp. 321.

Sheard, J., D'Souza, D., Lopez, M., Luxton-Reilly, A., Putro, I.H., Robbins, P., Teague, D. & Whalley, J.L. 2015, "How (not) to write an introductory programming exam", .

Simon, Chinn, D., de Raadt, M., Philpott, A., Sheard, J., Laakso, M., D'Souza, D., Skene, J., Carbone, A., Clear, T., Lister, R. & Warburton, G. 2012, "Introductory Programming: Examining the Exams", *Proceedings of the Fourteenth Australasian Computing Education Conference - Volume 123*Australian Computer Society, Inc, Darlinghurst, Australia, Australia, pp. 61.

Sprint, G. & Cook, D. 2015, "Enhancing the CS1 student experience with gamification", , pp. 94.

Sweller, J. 1994, "Cognitive load theory, learning difficulty, and instructional design", *Learning and instruction,* vol. 4, no. 4, pp. 295-312.

T. Rajala, E. Kaila, J. Holvitie, R. Haavisto, M. J. Laakso & T. Salakoski 2011, "Comparing the collaborative and independent viewing of program visualizations", *2011 Frontiers in Education Conference (FIE)*, pp. F3G-1.

T. Rajala, T. Salakoski, E. Kaila & M. J. Laakso 2010, "How does collaboration affect algorithm learning? A case study using TRAKLA2 algorithm visualization tool", *2010 2nd International Conference on Education Technology and Computer*, pp. V3-504.

T. Richter & D. Boehringer 2014, "Towards electronic exams in undergraduate engineering", *2014 IEEE Global Engineering Education Conference (EDUCON)*, pp. 196.

T. S. Indi 2016, "An Experience Report of Flipped Classroom Strategy Implementation for Java Programming Course", *2016 IEEE Eighth International Conference on Technology for Education (T4E)*, pp. 240.

Thomas, P., Price, B., Paine, C. & Richards, M. "Remote electronic examinations: student experiences", *British Journal of Educational Technology,* vol. 33, no. 5, pp. 537-549.

Titan, Ferdianto, G. G. F. P. Desak & Lena 2017, "A comparative study of teaching styles in online learning environment", *2017 International Conference on Information Management and Technology (ICIMTech)*, pp. 25.

Traynor, D., Bergin, S. & Gibson, J.P. 2006, "Automated Assessment in CS1", *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*Australian Computer Society, Inc, Darlinghurst, Australia, Australia, pp. 223.

V. Karavirta, R. Haavisto, E. Kaila, M. J. Laakso, T. Rajala & T. Salakoski 2015, "Interactive Learning Content for Introductory Computer Science Course Using the ViLLE Exercise Framework", *2015 International Conference on Learning and Teaching in Computing and Engineering*, pp. 9.

Walker, H.M. 2011, "How to Challenge Students", *ACM Inroads,* vol. 2, no. 3, pp. 14-15.

Watson, C. & Li, F.W.B. 2014, "Failure Rates in Introductory Programming Revisited", *Proceedings of the 2014 Conference on Innovation \&\#38; Technology in Computer Science Education*ACM, New York, NY, USA, pp. 39.

Wheeler, S., YEoMAnS, P. & WHEElER, D. 2008, "The good, the bad and the wiki: Evaluating student-generated content for collaborative learning", *British journal of educational technology,* vol. 39, no. 6, pp. 987-995.

Wibowo, S., Grandhi, S., Chugh, R. & Sawir, E. 2016, "A Pilot Study of an Electronic Exam System at an Australian University", *Journal of Educational Technology Systems,* vol. 45, no. 1, pp. 5-33.

Vihavainen, A., Airaksinen, J. & Watson, C. 2014, "A systematic review of approaches for teaching introductory programming and their influence on success", *Proceedings of the tenth annual conference on International computing education research*ACM, , pp. 19.

Winslow, L.E. 1996, "Programming pedagogy—a psychological overview", *ACM Sigcse Bulletin,* vol. 28, no. 3, pp. 17-22.

Wulf, T. 2005, "Constructivist Approaches for Teaching Computer Programming", *Proceedings of the 6th Conference on Information Technology Education*ACM, New York, NY, USA, pp. 245.

Yonglei Tao & J. Nandigam 2016, "Programming case studies as context for active learning activities in the classroom", *2016 IEEE Frontiers in Education Conference (FIE)*, pp. 1.