Vapaaehtoisen harjoitustentin vaikutus ohjelmoinnin johdantokurssin lopputulokseen

Minna Ikiviita 56611
Turun yliopisto
Tulevaisuuden teknologioiden laitos
Tietojenkäsittelytiede
Pro gradu -tutkielma
Lokakuu 2019

TURUN YLIOPISTO
Tulevaisuuden teknologioiden laitos

IKIVIITA, MINNA: Vapaaehtoisen harjoitustentin vaikutus ohjelmoinnin johdantokurssin lopputulokseen.

Pro gradu -tutkielma, 62 sivua, ei liitesivuja
Tietojenkäsittelytiede
Lokakuu 2019

TIIVISTELMÄ

Ohjelmoinnin opetusta on nykyisin jo joka kouluasteella. Viimeisimmän vuonna 2016 käyttöön otetun perusopetuksen opetussuunnitelman mukaan ohjelmointia opetetaan jo perusasteen 1. luokalta lähtien. Ohjelmoinnin opiskelu ja oppiminen on kuitenkin ollut perinteisesti vaikeaa. Tarkkaa syytä tähän ei ole tiedossa, mutta syiksi on esitetty muun muassa ohjelmoinnin vaatimia monimutkaisia kognitiivisia taitoja, ongelmanratkaisutaitoja ja motivaatiota.

Ohjelmointia tentitään nykyään lähes poikkeuksetta sähköisillä tenteillä. Sähköisiä tenttejä voidaan järjestää usealla eri alustalla (esimerkiksi Moodle, Exam ja ViLLE) ja eri tavoin. Tässä työssä esitellään erityisesti Turun yliopiston Tulevaisuuden teknologioiden laitoksella kehitetty ViLLE-oppimisympäristö. ViLLE mahdollistaa ohjelmoinnin opetuksen, automaattisen arvioinnin ja myös ohjelmoinnin sähköisen tenttimisen.

Työssä tutkittiin vapaaehtoisen harjoitustentin vaikutusta kurssin lopputulokseen kolmella ohjelmoinnin peruskurssilla syksyltä 2017. Kursseilla oli eri kohderyhmät, mutta pääsääntöisesti sama rakenteellinen sisältö. Kaikilla kursseilla oli vapaaehtoinen harjoitustentti, josta ei saanut pisteitä kurssin kokonaispistemäärään. Harjoitustentillä tarkoitetaan tenttiä, joka vastaa mahdollisimman tarkkaan lopputenttiä kysymysten muodoilta ja tenttitavaltaan.

Vapaaehtoisen harjoitustentin suoritti kaikilla kolmella kurssilla yli ¾ kurssin tenttiosallistujista. Harjoitustentti vaikutti ristiriitaisesti eri kursseilla kurssin tenttipistemäärään ja kokonaispistemäärään. Vain yhdellä kurssilla positiivinen korrelaatio oli vahva. Kuitenkin jokaisella tutkittavalla kurssilla harjoitustentin tehneet saivat mediaanituloksissa paremmat pisteet sekä lopputentissä ja kurssin kokonaispisteessä kuin ne, jotka eivät tehneet vapaaehtoista harjoitustenttiä.

Avainsanat: ohjelmointi, ohjelmoinnin opetus, verkko-oppimisympäristöt, sähköinen tentti, harjoitustentti.

UNIVERSITY OF TURKU
Department of Future Technologies

IKIVIITA, MINNA: Voluntary practice exam on introductory programming courses and its effect on the course result.

Master's Thesis, 62 pages, no appendices
Computer Science
October 2019

ABSTRACT

In Finland, programming is nowadays taught in every level of school. Learning programming has been traditionally difficult, and no exact reason has been found for that. Some theories are that programming requires such complex cognitive skills, advanced problem-solving skills and motivation.

For majority of programming courses electronic exams are used. There are several different platforms for electronic exams, for instance Moodle, Exam and ViLLE. This thesis focuses mostly on ViLLE, which has been developed in University of Turku. ViLLE enables teaching programming, automatic assessment and electronic exams using programming tools.

This thesis focuses on voluntary practice exam and its effect on the course result on three introductory programming courses taught in autumn 2017 at University of Turku. The target audiences for the courses are different, but the basic structure is the same. All three courses also had a voluntary practice exam which did not add points to the overall course points. Practice exams are exams which have the same type of structure and questions than the final exam.

The results of this study show over ¾ of the students taking the final exam also took the practice exam. The effect of the practice exam was contradictory in the courses to the points of the final exam and the overall course points – only one course had a strong positive correlation. The median results of the final exam and the overall course points were higher for the students who took the practice exam than for the students who did not take it.

Avainsanat: programming, teaching programming, web-based learning environments, electronic exam, practice exam.

Sisällys

1 Johdanto

”Opettajat ovat entistä täsmällisempiä määrittäessään, mitä heidän opetuksensa tulee antaa oppilaille. He alkavat myös ymmärtää, kuinka paljon hyötyä sekä oppilaille että opettajille on siitä, että on käytettävissä luotettavia mittausmenetelmiä, jotka osoittavat, mitkä osat oppilaille suunnattavista viesteistä pääsevät perille.” (Kay ym. 1970, 8.)

Näin kuvailevat Kay, Dodd ja Sime (1970) esipuheessaan silloin uutta opetusteknologiaa, joka vasta vähitellen saa jalansijaa opetuksen parissa: opetuskoneita ja tietokoneella ohjattua opetusta.

Ohjelmoinnin opetusta on nykyisin joka kouluasteella, ja sen opetukseen kiinnitetään yhä enemmän huomiota. Viimeisimpään perusopetuksen opetussuunnitelmaan, joka otettiin käyttöön porrastetusti alkaen vuodesta 2016, ohjelmoinnin opetus tuotiin näkyvästi (OPS 2014). Perusopetuksen eri vuosiluokilla ohjelmoinnilliseen ajatteluun tutustutaan vaiheittain ja eri oppiaineissa, muun muassa matematiikassa ja käsitöissä. Ylioppilaskirjoitusten siirryttyä kokonaan digitaalisiksi sähköinen tenttiminen on tuttua, ainakin otsikkotasolla, jokaiselle. Digitaaliset ylioppilaskokeet vaikuttavat myös lukioiden opiskeluun ja tenttitapoihin. Sähköisen tenttijärjestelmän käyttöä harjoitellaan opintojen alusta asti ja se on osa arkipäivää nykyajan opiskelijalle.

Ohjelmoinnin oppiminen on kuitenkin perinteisesti ollut hankalaa, koska se vaatii hyvää ongelmanratkaisukykyä, kykyä ymmärtää olemassa olevaa koodia ja valitun ohjelmointikielen syntaksin osaamista (Esteves ym., 2009, 5). Lisäksi ohjelmoinnin peruskurssit korkea-asteella ovat haasteellisia sekä opiskelijoille, että opettajille: peruskursseilla luodaan pohja, joka vaaditaan ohjelmoinnin oppimiseen, mutta jonka tietomäärä voi olla vaikea sisäistää. (Kay ym., 2000, 109-110).

Tämän työn tavoitteena on selvittää, miten ohjelmointia opitaan ja miten sitä opetetaan, erityisesti korkea-asteella. Tutkin kolmea aloituskurssia ohjelmointiin, jotka on pidetty syksyllä 2017 Turun yliopistossa. Näistä

kursseista tutkin erityisesti vapaaehtoisen harjoitustentin vaikutusta tenttipisteisiin lopullisessa tentissä.

Kiinnostuin aiheesta maisteriopintojeni syventävällä Learning Analytics -kurssilla tehdyn tutkimuksen vuoksi. Ryhmämme sai odottamattomia tuloksia harjoituskokeen merkityksestä, joten päätin jatkaa aiheen tutkimista. Tuloksiamme ei ole julkaistu, mutta työtä aloittaessamme oletimme, että harjoitustentillä olisi merkitystä lopputentin arvosanaan arvosanaa korottavasti. Kuitenkin silloisten tulostemme perusteella harjoitustentin suorittaneet saivat huonommin pisteitä lopputentissä. Harjoitustentin suorittaneilla oli enemmän pisteitä demonstraatioista ja muista harjoitustehtävistä. Kahden tutkittavan ryhmän välillä ei ollut merkittävää eroa alkutaidoissa. Tämän työn pohjalta halusin tutkia kysymystä myös pro gradu -tutkielmassani.

Ohjelmointikurssien tentit suoritetaan yleensä jonkinlaisena sähköisenä tenttinä. Sähköisiin tentteihin on monia eri vaihtoehtoja, esimerkiksi tentit erilaisissa verkko-oppimisympäristöissä (muun muassa ViLLE, Moodle) tai hyödyntäen oppilaitoksien sähköisiä tenttipalveluita, esimerkiksi Turun yliopistossa Exam. Valittu ympäristö vaikuttaa merkittävästi siihen, mitä tentissä voidaan kysyä ja miten.

Tutkielmani tutkimuskysymykset ovat:

1. Vaikuttaako ViLLE-oppimisympäristössä tehty vapaaehtoinen harjoitustentti ohjelmoinnin peruskurssin lopputentin pistemäärään?
2. Miten harjoitustentin vaikutus eroaa tutkittavien ryhmien välillä (IT-pääaineopiskelijat, sivuaineopiskelijat, avoimen yliopiston opiskelijat)?

Tässä työssä tutkittavilla kolmella ohjelmoinnin kursseilla on ollut käytössä vapaaehtoinen harjoitustentti. Tentti on sähköinen ja toteutettu ViLLE-verkko-oppimisympäristössä. Jokaisella kurssilla on ollut samantyyppinen sisältö, mutta hyvin erilaiset kohderyhmät. Tästä syystä tutkimuksen tarkoituksena on selvittää myös, vaikuttaako harjoitustentti eri tavalla näiden kolmen eri kohderyhmän kohdalla.

Luvussa 2 selvitetään ohjelmoinnin opiskelua, miten ohjelmointia opitaan, millaisia ongelmia ohjelmoinnin oppimisessa on ja mitä erilaisia metodiikkoja ohjelmoinnin oppimiseen on käytössä. Lisäksi otetaan katsaus arviointiin.

Luvussa 3 selvitetään, mitä tarkoitetaan verkko-oppimisympäristöllä ja millainen rooli opettajalla siinä on. Luvun 3 aliluvuissa selvitetään myös sähköisen tentin ja harjoitustentin käsitteitä.

Luvussa 4 tutustutaan ViLLE verkko-oppimisympäristöön ja siellä erityisesti, miten sähköinen tentti voidaan ViLLEssä järjestää.

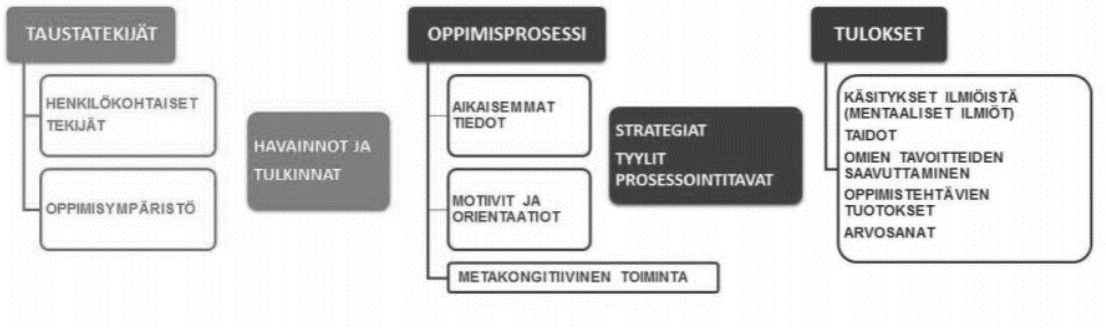
Luvussa 5 avataan työn tutkimuskysymykset, menetelmät ja rajaukset. Luvussa 6 kerron tutkimukseni tulokset. Luvussa 7 kerron tuloksista vetämäni johtopäätökset. Luku 8 on yhteenveto.

2 Ohjelmoinnin opiskelu

Oppiminen on moniulotteinen käsite, josta jokaisella on käsityksiä ja uskomuksia, vaikka niitä ei olisi tietoisesti pohtinut. Marlene Schommerin (1990) mukaan omat uskomukset tiedosta ja oppimisesta voivat olla mustavalkoisia, ja vaikuttaa menestykseen koulussa, myös haitallisesti.

Haastattelututkimusten perusteella (Tynjälä 1999, 12) on tutkittu ihmisten arkikäsityksiä oppimisesta ja tulosten perusteella oppiminen voidaan jakaa kahteen pääryhmään: oppiminen toistavana toimintona ja oppiminen transformaationa. Oppiminen toistavana toimintona tarkoittaa tietojen lisääntymistä, muistamista ja soveltamista. Oppiminen transformaationa taas on kehittymisenä tai muuttumisen oppijan ajattelussa tai toiminnassa.

Tynjälä (1999, 16-20) esittelee oppimisen kokonaismallin (kuva 1), jossa on kolme rakenneosaa: taustatekijät, oppimisprosessi ja tulokset.



Kuva 1 Oppimisen kokonaismalli (Tynjälä 1999, 17).

Opettaessa, opeteltaessa ja oppiessa ohjelmointia voidaan myös soveltaa Tynjälän oppimisen kokonaismallia. Ohjelmoinnin opetuksessa taustatekijöillä (muun muassa luokka-aste) on merkitys esimerkiksi opetusmenetelmän tai oppimisympäristön valintaan. Samoin aiemmat kokemukset vaikuttavat siihen, millaisia havaintoja ja tulkintoja on mahdollista tehdä oppimateriaalista. Oppimisprosessi voi myös olla myös alkuvaiheessa hitaampaa aloittelijoilla kuin

jo kokeneilla ohjelmoinnin opiskelijoilla. Erilaiset motivaatiotekijät, kuten esimerkiksi syyt miksi opiskelija haluaa oppia ohjelmointia, vaikuttavat merkittävästi myös oppimisen tuloksiin (katso esimerkiksi Vihavainen 2014.)

Luvussa 2.1 selvitän, millaisia taitoja ohjelmointiin tarvitaan ja miten ohjelmointia opitaan. Luku 2.2 painottuu ohjelmoinnin oppimisen ongelmiin. Luvussa 2.3 esittelen ohjelmoinnin metodiikkaa ja kaksi tällä hetkellä suosiossa olevaa pedagogiikkaa, ongelmalähtöisen oppimisen (Problem Based Learning) ja käänteisen opetuksen (Flipped Learning). Luvussa 2.4 perehdytään arviointiin.

2.1 Miten ohjelmointia opitaan?

Ohjelmoinnin opiskeluun ja pedagogiaan on kiinnitetty huomiota maailmanlaajuisesti, mutta Watsonin & Lin (2014) mukaan kurssien läpäisy ei ole siitä huolimatta parantunut merkittävästi verrattuna aiempiin tuloksiin. Lisäksi on huomattu, ettei kurssien läpäisyyn vaikuta merkittävästi pedagogisesti ulkopuoliset tekijät, kuten kurssilla käytettävä ohjelmointikieli.

Vihavainen ym. (2014) tutkimuksen mukaan kuitenkin ohjelmoinnin kurssien läpäisy on keskimäärin kolmasosan korkeampi kursseilla, joilla on siirrytty pois perinteisestä opettajavetoisesta teorialuennot ja demonstraatiot -lähestymistavasta, ja käytetään jotain muuta opetusmenetelmää.

Toisin sanoen ohjelmointia opitaan ohjelmoimalla, eikä puhumalla ohjelmoinnista tai kuuntelemalla teoriaa. Pelkän syntaksin ja muun ohjelmointiin liittyvän tiedon opettaminen ei riitä, vaan tarvitaan käytännön harjoittelua ohjelmoinnissa vaadittavista ajatusmalleista, jotta opiskelija osaa yhdistää teorian käytäntöön.

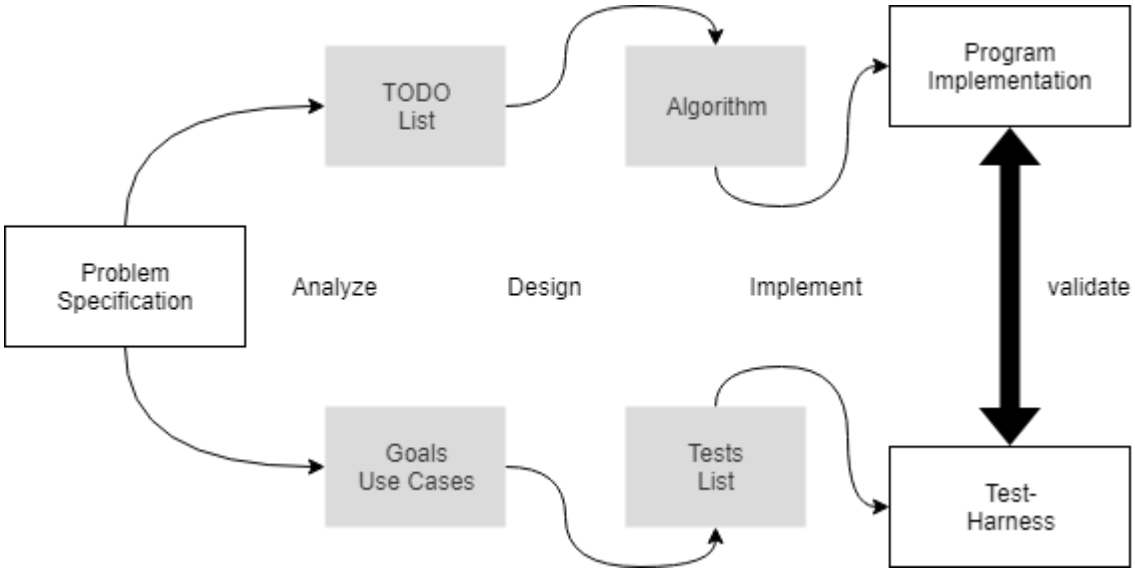
Gasparin ym. (2009, 264-265) mukaan tarvittavat taidot ohjelmointiin ovat:

1. Ohjelman suunnittelu- ja toteuttamistaidot (Program design and implementing skills)
2. Ohjelman testaustaidot (Program testing skills)
3. Ongelmanratkaisutaidot (Troubleshooting skills)

Ohjelman suunnittelu- ja toteuttamistaidot hankitaan siten, että ymmärretään ohjelmoinnin pyrkivän olemaan luova tapa ratkaista yksityiskohtaisesti määritelty ongelma, algoritmi. Oleellista ohjelmoinnin opetuksessa on sisäistää, miten ongelma voidaan ratkaista, eikä vain opetella ratkaisua ulkoa. Ulkoa opettelun vaarana on opiskelijan kyvyttömyys soveltaa ratkaisua hieman erilaisen ongelman ratkaisussa. (Gaspar ym. 2009, 264-265.)

Ohjelman testaustaidot ovat oleellisia ohjelmoinnissa, myös aloittelevalle ohjelmoijalle. Jos ohjelmaa ei osaa testata, sen validointi ja oikeellisuus voi jäädä puutteelliseksi. Testaus pitää myös tehdä vaadittuja määrittelyjä vasten, eikä puutteellisella testauksella opiskelijalla ole tietoa, toimiiko ohjelma vaadittujen määrittelyjen mukaisesti. (Gaspar ym. 2009, 265.)

Ongelmanratkaisutaidot kattavat ohjelmoinnissa seuraavia asioita: opiskelijan kyky tunnistaa virheitä ohjelmassa ja sen jälkeen kyky löytää virhe ohjelmakoodista. Ongelmanratkaisutaidot ovat sidoksissa kahden aiemman määritellyn taidon – suunnittelu- ja toteuttamistaitojen ja testaamisen – kanssa. (Gaspar ym. 2009, 265.) Ilman ongelmanratkaisutaitoja opiskelija ei voi kehittyä ohjelmoinnissa, koska hänellä ei ole kykyä korjata tai löytää virheitä ohjelmastaan.



Kuva 2 Ohjelmoinnin opettamisen eri prosesseja johdantokursseilla (Gaspar ym., 2009, 264).

Kuvassa 2 on kaksi eri mallia, miten esimerkiksi on mahdollista opettaa ohjelmointia johdantokursseilla. Ratkaistava ongelma on molemmissa sama ja lopputuloksissa voidaan hyödyntää molempien tapojen tuloksia. Lopputulokseen päästään samojen vaiheiden kautta analysoinnin, suunnittelun ja implementoinnin kautta, mutta käytettävät työkalut ovat erilaiset. Kuitenkin molemmat lopputulokset ovat sellaisia, että toisella voidaan validoida toisen lopputulos.

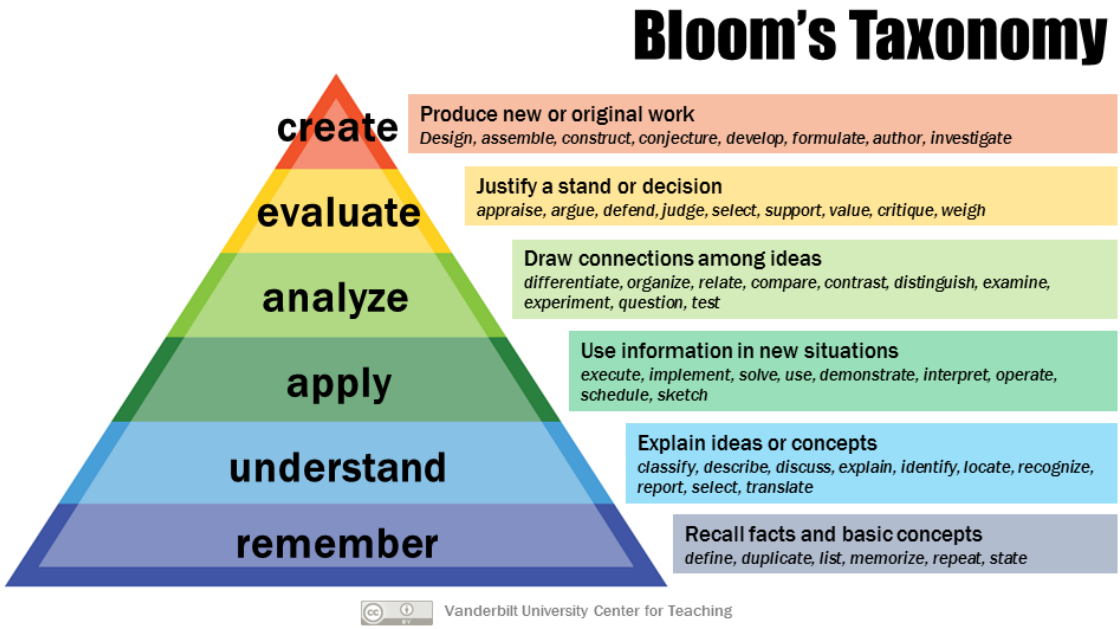
Vuonna 1956 kehitetty Bloomin taksonomia jakaa oppimisen ja osaamisen eri kategorioihin. Olennaista ohjelmoinnin oppimisessa on taksonomian kognitiiviset kategoriat, jotka Scott (2003) määritteli seuraavasti:

1. Tieto / Tiedon muistaminen (Knowledge / Recall of Data): Oppija muistaa ulkoa opettelemansa käsitteet, kuten muuttujien tai tietorakenteiden tyypit.
2. Ymmärtäminen (Comprehension): Oppija ymmärtää ja pystyy tulkitsemaan esimerkiksi annetusta koodista, mitä ohjelmassa tapahtuu.
3. Soveltaminen (Application): Oppija pystyy soveltamaan annettua tehtävää toisenlaiseksi, esimerkiksi muokkaamaan koodissa olevan toistorakenteen toiseen vastaavaan.

4. Analysointi (Analysis): Oppija osaa analysoida esimerkiksi kahden samantyyppisen järjestelmän etuja ja haittoja.
5. Yhdistäminen (Synthesis): Oppija osaa aiemmin oppimansa perusteella luoda uutta, esimerkiksi jatkoa olemassa olevalle ohjelmalle.
6. Arviointi (Evaluation): Oppija pystyy arvioimaan kriittisesti annettua ratkaisua, esimerkiksi löytämään ohjelman lähdekoodista loogisen virheen.

Jokainen taso edellyttää edellisen tason oppimista: päästäkseen esimerkiksi tasolle 5 on osattava myös tasot 1-4. Mitä ylemmällä tasolla oppija on, sitä monimutkaisempia ongelmia hän osaa ratkaista, ja toisaalta sitä hienostuneempia ratkaisumalleja ratkaisuissa voidaan käyttää.

Bloomin taksonomiasta julkaistiin vuonna 2001 uudistettu versio, jossa mallia tarkennettiin ja siihen lisättiin tasoja. Alkuperäisen mallin asiasanat olivat substantiiveja, mutta uudistettuun malliin asiasanat tuotiin verbimuodossa (katso kuva 3). Muutoksella haluttiin tarkentaa uuden version dynaamisuutta alkuperäisen version staattisuuteen. (Bloom ym. 2001, 263-268.)



Kuva 3 Bloomin taksonomian uudistettu versio (Vanderbilt University Center for Teaching).

Vertailemalla alkuperäistä ja uudistettua Bloomin taksonomiaa, huomataan, että perusajatus ja tasot ovat pääsääntöisesti pysyneet samana. Kaiken perustana on edelleen tiedon muistaminen ja ymmärtäminen, siitä jatkaen soveltamiseen ja analysointiin. Kahden ylimmän tason järjestys on muuttunut (alkuperäisessä Synthesis – Evalution, uudistetussa Evaluate – Create), uuden luominen uudistetussa versiossa ylimpänä.

Ohjelmointia opiskellaan yliopistotasolla isoissa ryhmissä, ja näiden ryhmien sisällä opiskelijat ovat eri tasoilla osaamisensa kanssa Bloomin taksonomian mukaan, peruskursseillakin. Haasteena ohjelmoinnin kursseilla onkin, miten luoda materiaalia, joka on riittävän haastavaa myös muille kuin alimmilla tasoilla oleville. Jos materiaali on liian helppoa, ylemmillä tasoilla oleva opiskelija turhautuu. Jos taas materiaali on haastavampaa ja sopivaa ylemmille tasoille, niin alemmilla tasoilla oleva opiskelija turhautuu, koska ei osaa, eikä siten myöskään todennäköisesti opi. Kokemukseni mukaan opiskelun eriyttäminen eri taitotasoille on ohjelmoinnin opettamisessakin ongelma, erityisesti jos ryhmäkoot ovat isoja.

Skeemat eli sisäiset mallit ovat kaikessa oppimisessa tärkeitä. Skeemojen avulla ihminen voi joustavasti toimia ja havainnoida vaihtelevissa tilanteissa. Ulric Neisser määritteli skeemat osaksi havaintosykliä. (Lehtinen ym. 2016, 98-100.)

Neisserin havaintosyklissä on kolme osaa, joista skeemat ovat havaitsijan sisäinen osa havaintosyklistä. Kuvassa 4 on mukailtu Neisserin kehittämää havaintosykliä. Syklissä ihmisen sisäiset mallit, skeemat, ohjaavat havainnointia. Havainnoinnista ihminen saa tietoa. Tämä saatu tieto muokkaa tai vahvistaa olemassa olevaa sisäistä mallia. Sykli on jatkuva ja ohjaa ihmisen oppimista. (Neisser 1976, 20-24.)

Kuva 4 Ulric Neisserin havaintosykli.

2.2 Ohjelmoinnin oppimisen ongelmat

Ei ole tarkkaan tiedossa, miksi ohjelmointi on vaikeaa. Asiasta on lukuisia erilaisia teorioita (katso esimerkiksi johdannossakin jo mainitut Kay ym. 2000 ja Esteves ym. 2009). Kuten jo edellisen luvun Gasparin ym. (2009) määrittelystä huomattiin, ohjelmointi vaatii useampia taitoja, eikä vain yhdellä taidolla pärjää. Ohjelmointiongelman ratkaisuun tarvitaan monimutkaisia kognitiivisia taitoja samanaikaisesti ratkaistakseen käsillä olevan ongelman

Ohjelmoinnin oppimisen vaikeudeksi on muun muassa esitetty ohjelmoinnin luontaista vaikeutta. Ohjelmoidessa käytetään useita kognitiivisesti vaikeita taitoja, jossa opiskelijan on osattava samanaikaisesti rakentaa ja hyödyntää näitä taitoja ratkaistakseen ohjelmointiongelman. Toisaalta ohjelmoinnin oppimisen vaikeuden syiksi on annettu opiskelijoiden ominaisuuksia: motivaation puute tai kyvyttömyys luoda ohjelmoinnissa vaadittavia ajatusmalleja siitä, miten ohjelmat kytkeytyvät toisiinsa.

Erityisesti ohjelmoinnin peruskursseille tuleville opiskelijoilla on myös paljon eroja lähtötason kanssa: osa opiskelijoista on ohjelmoinut yhden tai useamman ohjelmointikielen kanssa ja toiset ottavat kurssilla ensikosketuksen ohjelmointiin. Tämä aiheuttaa haasteita myös opettajalle, erityisesti perinteisellä luentokurssilla. Perinteisellä luentokurssilla tarkoitetaan luentopohjaista opetusta, jossa opetetaan staattisilla materiaaleilla dynaamista asiaa, eli teorian ja syntaksin painotus on suuri. (Esteves ym., 2009, 5-6.)

Ohjelmoinnin vaikeuden perimmäiseksi syyksi on useissa tutkimuksissa huomattu puutteet ongelmanratkaisukyvyssä. Samalla ongelmanratkaisukyvyn on huomattu olevan yksi tärkeimmistä ominaisuuksista ohjelmoinnin oppimisessa. Näiden tutkimusten pohjalta onkin suositeltu ongelmanratkaisukyvyn kehittämisen olevan yksi tärkeimmistä taidoista, joihin kiinnitettäisiin huomiota, kun suunnitellaan pedagogisesti ohjelmoinnin peruskursseja. (Apiola 2013, 8.)

2.3 Ohjelmoinnin opettaminen / metodiikka

Ohjelmoinnin opetuksessa voidaan hyödyntää useita pedagogisia malleja. Tässä luvussa käsitellään lyhyesti yleisesti konstruktivistista oppimiskäsitystä. Pedagogisista malleista tutustutaan ongelmalähtöiseen oppimiseen (Problem Based Learning) ja käänteiseen oppimiseen (Flipped Learning).

Konstruktivismi on yksi oppimiskäsityksistä, malleista, joiden mukaan oppimista tapahtuu. Konstruktivistisen käsityksen mukaan oppimista tapahtuu jo ympäristöä havainnoimalla. Ympäristön havainnointia ohjaavat aikaisemmin opitut tiedot ja uskomukset, ja ne ohjaavat aktiivista oppimisprosessia. (Rinne, Kivirauma & Lehtinen 2010, 30-31.)

Konstruktivismilla on kaksi eri suuntausta: yksilökeskeinen konstruktivismi ja sosiaalinen konstruktivismi / sosiokonstruktivismi. Yksilökeskeisessä konstruktivismissa korostetaan havaintojen tulkitsemista sisäisten rakenteiden pohjalta. Sosiokontruktivismissa taas korostetaan ihmisen toiminnan ja kielen

merkitystä oppimisessa. Konstruktivismi käsitetään kuitenkin yleiskäsitteenä, joka sisältää molemmat mainitut lähestymistavat. (Tynjälä 1999, 26-27.)

Konstruktivismilla on konkreettisia käytännön merkityksiä käytännön pedagogiikalle. Tynjälä (1999, 60-67) listaa teoksessaan seuraavat:

1. Oppijan aktiivisuuden merkitys ja opettajan roolin muuttuminen
2. Oppijan aikaisemmat tiedot
3. Metakognitiivisten taitojen kehittäminen
4. Ymmärtäminen on tärkeämpää kuin ulkoa osaaminen
5. Erilaisten tulkintojen huomioon ottaminen
6. Faktapainotteisuudesta ongelmakeskeisyyteen
7. Oppimisen tilannesidonnaisuuden huomioon ottaminen
8. Monipuolisten representaatioiden kehittäminen
9. Sosiaalisen vuorovaikutuksen painottaminen
10. Uusien arviointimenetelmien kehittäminen
11. Tiedon suhteellisuuden ja tuottamistapojen esiin tuominen
12. Opetussuunnitelmien kehittäminen

Näistä useampia voidaan pitää merkittävinä myös ohjelmoinnin opetuksessa. Erityisesti kohdat 1-6 ovat tässäkin työssä esitettyjen lähteiden perusteella juuri ohjelmoinnin opettamisessa ja oppimisessa tärkeitä. Enää ohjelmointia ei opita kalvosulkeisten avulla, vaan oppijan aikaisemmilla taidoilla on suuri merkitys oppimisnopeuteen, ja tärkeämpää on opettaa metakognitiivisia taitoja kuin esimerkiksi jonkun yksittäisen ohjelmointikielen syntaksia. Lisäksi ohjelmoinnissa on erittäin tärkeää ymmärtää, miten koodi muodostuu kuin vain opetella ulkoa valmiita koodinpätkiä.

Vihavaisen ym. (2014, 22-26) artikkelissa tutkittiin ohjelmoinnin peruskursseja ja niissä käytettyjä malleja. Tutkittavia artikkeleita oli 32 ja ne sisälsivät 13 erilaista mallia, joita käytettiin ohjelmoinnin peruskursseilla. Tutkimuksen perusteella ei löydetty yhtä ainoaa mallia, joka toimisi parhaiten ohjelmoinnin opetuksessa. Erot eivät kuitenkaan olleet yksittäisten mallien välillä tilastollisesti merkittäviä kursseilla, joita tutkittiin. Tutkimuksessa huomattiin

kuitenkin, että yhdistämällä eri malleja saatiin parempia tuloksia kuin pelkkää yhtä mallia käyttämällä.

Erkki Kaila esitteli väitöskirjassaan (2018, 56-59) ohjelmoinnin ja tietotekniikan peruskurssien opettamiseen mallin, jossa opetusteknologiaa ja verkko-oppimisympäristöjä hyödyntävässä opetuksessa on otettava huomioon viisi eri tekijää. Näitä tekijöitä ovat aktiivinen oppiminen ja jatkuva arviointi, heterogeeniset tehtävätyypit, sähköinen tentti, tutoriaalipohjainen oppiminen sekä jatkuva palaute. Väitöskirjan tutkimuksen pohjana on käytetty tässäkin työssä tutkittavaa ViLLE-oppimisjärjestelmää, josta lisää luvussa 4. Kailan tutkimuksen tulosten perusteella suunniteltu malli paransi opiskelijoiden arvosanoja ja kurssin läpäisyä tilastollisesti merkittävästi.

Seuraavissa aliluvuissa selvitetään, mitä on ongelmalähtöinen oppiminen ja käänteinen oppiminen. Kummassakin pedagogisessa mallissa opettajan rooli poikkeaa perinteisestä opettajavetoisesta roolista enemmän tuutorin / fasilitaattorin / ohjaajan rooliin. Opettaja on läsnä opetuksessa ja merkittävässä osassa tehtävien luomisessa, mutta oppiminen oppilaskeskeisempää.

2.3.1 Ongelmalähtöinen oppiminen

Problem Based Learning (jatkossa PBL) eli ongelmaperustainen tai ongelmalähtöinen oppiminen on oppijakeskeinen pedagoginen malli toisin kuin perinteiset opettajalähtöiset pedagogiset mallit. PBL kehitettiin yli 30 vuotta sitten lääketieteen opetukseen, mutta sittemmin metodia on muokattu ja laajennettu sekä alemmille kouluasteille, että eri aloille, joista yhtenä esimerkkinä matemaattiset aineet. PBL:n avulla opiskellaan pienissä ryhmissä, jotka tarvitsevat aina fasilitaattorin tai tuutorin ohjaamaan prosessia. Ryhmä ratkaisee ongelmaa tai ongelmia, joiden pitää olla jollain tavalla ”viallisia”, jotta malli toimii onnistuneesti ja mukailee tosielämän ongelmia. (Savery 2015, 5-9, 13.)

PBL:n avulla oppijat / opiskelijat kehittävät toimivan ratkaisun määriteltyyn ongelmaan käyttämällä apuna tiedon tutkimista, soveltamalla teoriaa ja käytäntöä sekä hyödyntämällä olemassa olevia tietoja ja taitoja. PBL:ssä tärkeää on, että opiskelijat ottavat itse vastuun oppimisestaan. Myös oppimisen analysointi ja itse- ja vertaisarviointi on avainasemassa. Erilaisia analysointi- ja arviointimenetelmiä käytetään sekä harjoituksen aikana että sen jälkeen, sekä yksilön että ryhmän oppimiseen ja tavoitteisiin. (Savery, 2015, 8-9.)

PBL:stä löytyy useita case-tutkimuksia, joissa sitä on hyödynnetty nimenomaan ohjelmoinnin ja tietojenkäsittelyn peruskursseilla (katso esimerkiksi Kay ym. 2000, Esteves ym. 2009). Näissä tutkimuksissa PBL:stä on löydetty sekä haittoja että hyötyjä ohjelmoinnin opetukseen. PBL vaatii enemmän resursseja kuin perinteiset opettajakeskeisesti opetetut kurssit: kurssien suunnittelu ja muokkaaminen vaatii opettajilta enemmän aikaa ja lisäksi ryhmä vaatii aina tuutorin tai fasilitaattorin ohjaamaan ongelman ratkaisua.

PBL:stä on kuitenkin lukuisia hyviä puolia: tutkituissa caseissa opiskelijoiden tulokset paranivat aiempiin opettajakeskeisiin kursseihin verrattuna. Opiskelijat myös kehittyivät kommunikointitaidoissa, yhteistyötaidoissa, kriittisessä ajattelussa ja ongelmanratkaisussa. Toisin sanoen taitoja, joita tulevat työnantajat arvostavat.

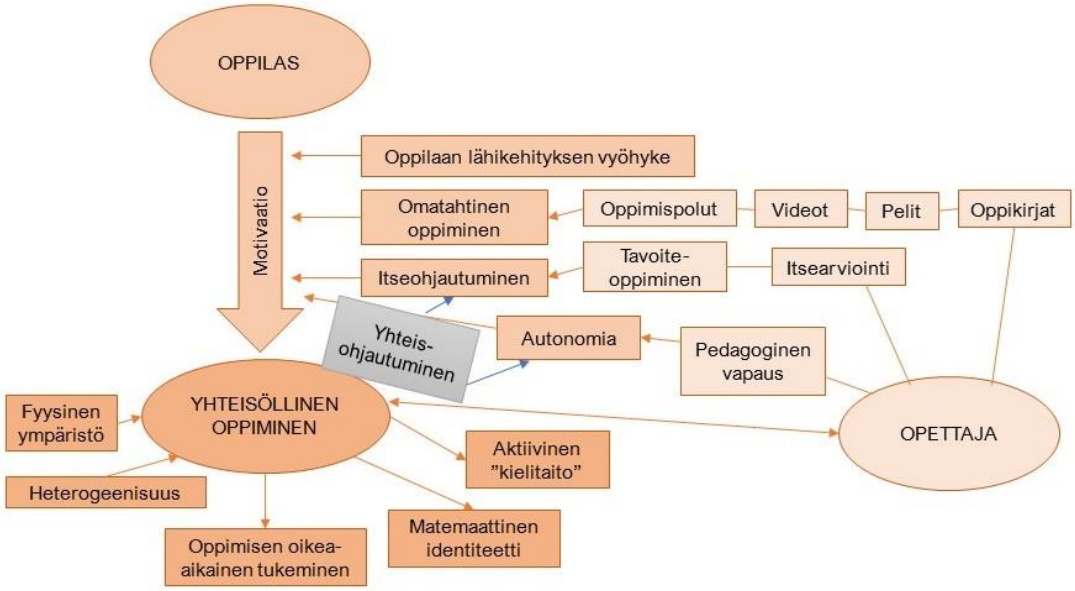
2.3.2 Käänteinen oppiminen

Flipped learning eli käänteinen oppiminen on viime vuosien uusi, keskustelua herättänyt menetelmä opetuksen malleista puhuttaessa. Yhdessä ilmiöoppimisen kanssa nämä kaksi suuntausta nähdään opetuksen tulevaisuutena. Suomessa käänteisen oppimisen pioneeri on Marika Toivola, jonka työhön tässäkin työssä tutustutaan.

Puhuttaessa käänteisestä oppimisesta käsitteissä pitää olla tarkkana. Käänteinen oppiminen (flipped learning) ja käänteinen opetus (flipped classroom) koetaan synonyymeina, mutta ne eivät sitä ole. Käänteinen opetus

on opetusmetodi, jossa tyypillisesti – ei kuitenkaan kategorisesti aina – teoriaan tutustutaan kotona ja ns. kotitehtävät tehdään oppitunneilla. Käänteinen oppiminen on sen sijaan opetusideologia, jossa opettajan rooli on totuttaa oppilaat omaehtoiseen ja oma-aloitteeseen oppimiseen. (Toivola ym. 2017, 20-21.)

Käänteinen oppiminen on globaalisti suosittua, mutta sillä ei vielä ole yleisesti hyväksyttyä teoreettista viitekehystä. Kuvassa 5 on Toivolan kehittämä käänteisen oppimisen pedagoginen malli, jossa toimijoina on oppilaan ja opettajan lisäksi yhteisöllinen oppiminen. Malli jakaantuu kolmeen osaan: oppilas yksilönä, yhteisö oppimisen mahdollistajana ja opettajan toiminta. Näillä kaikilla kolmella osakokonaisuudella on oma roolinsa käänteisessä oppimisessa. (Toivola ym. 2017, 22-23.)



Kuva 5 Käänteisen oppimisen pedagoginen malli (Toivola ym. 2017, 23).

Käänteisen opetuksen hyötyjä on oppilaan sisäisen, autonomisen motivaation lisääntyminen. Oppiminen ei ole enää näennäistä, vain kurssikoetta varten tehtävää, vaan oppilas opiskelee oppimaan omatahtisesti ja itseohjautuvasti.

Opettajan rooli on tukea tätä kehitystä ja varmistaa, että jokaiselle oppilaalle löytyy omalla tasollaan olevaa oppimateriaalia.

Käänteinen opetus voi olla aluksi hankalaa sekä oppilaille, että heidän vanhemmilleen (kun puhutaan käänteisestä opetuksesta perusasteella), koska näennäinen edistyminen esimerkiksi muistiinpanojen osalta ei ole samanlaista kuin ennen. Oppilaat saavat edetä omaan tahtiinsa. Toisin sanoen hitaammat oppilaat saavat keskittyä perustehtäviin, mutta edistyneet oppilaat saavat omalle tasolleen sopivia tehtäviä. Oppilaita myös kannustetaan toimimaan ryhmissä, ratkaisemaan ongelmia yhdessä sekä kysymään opettajalta apua tarvittaessa. Virheitä ei pidä hävetä, eikä piilotella, vaan ne nähdään oppimisen kannalta merkittävinä, hyvinä asioina. Koska menetelmä on oppilaskeskeinen, eli poikkeaa vahvasti perinteisestä opettajakeskeisistä menetelmistä, vie menetelmään tottuminen ja vastuun ottaminen omasta työstä aikaa.

Käänteistä opetusta ohjelmoinnin opetuksessa on tutkittu jonkin verran. Esittelen seuraavaksi kaksi artikkelia, joissa on hyödynnetty käänteistä opetusta. Erityisesti minua kiinnosti, miten käänteistä opetusta on hyödynnetty korkea-asteella, koska suomalaiset tutkimukset käänteiseen opetukseen ovat vahvasti perusasteelta ja / tai pro gradu -tutkielmia (katso esimerkiksi Toivola ym. 2017, Toivola 2019).

Håkan Jonssonin artikkelissa raportoitiin kokeilusta, jossa käänteistä opetusta tutkittiin olio-ohjelmoinnin ja suunnittelun kurssilla ensimmäisen vuoden insinööriopiskelijoille. Kurssilla oli jo vuosia ollut sama, perinteinen rakenne: luennot, pakolliset ohjelmointitehtävät ja kurssin lopuksi kirjallinen koe. Kokeilussa luentojen tilalle vaihdettiin käänteisen opetuksen malliin luokkaopetusta, johon yhdistettiin vertaiskeskustelu ja opetushetket (Just-in-time teaching). Kurssilla otettiin myös käyttöön web-pohjainen suurelle massalle tarkoitettu verkko-oppimisympäristö. (Jonsson 2015.)

Jotta kurssin opetustyylin vaikutusta loppuarvosanoihin pystyttiin arvioimaan, kurssin osallistujat jaettiin kahteen ryhmään. 70 opiskelijan ryhmä suoritti kurssin uudella tavalla ja 57 opiskelijan kontrolliryhmä suoritti kurssin vanhalla

tavalla. Kurssin kirjallinen loppukoe oli molemmille ryhmille sama, ja se arvioitiin samalla tavalla. Uudella tavalla kurssin suorittaneista 81 % läpäisi loppukokeen, kun taas vanhalla tavalla kurssin suorittaneista vain 60 % läpäisi kokeen. Artikkelissa ei tutkittu pelkältään kurssin läpäisyä, vaan myös millä arvosanoilla kurssi oli läpäisty. Uudella tavalla kurssin suorittaneista 58 % sai hyvän arvosanan, kun taas kontrolliryhmässä hyvän arvosanan sai 32 %. (Jonsson 2015.)

Toinen artikkeli (Maher, Latulipe, Lipford & Rorrer 2015) keskittyy käytäntöihin neljän yliopistotason tietotekniikan / tietojenkäsittelytieteen kurssista, jotka muutettiin käyttämään metodina käänteistä oppimista (kutsutaan myös ”flippaamiseksi”). Kurssit muutettiin kahden vuoden aikana (syksy 2012 – kevät 2014), jolloin ne ehdittiin järjestää useampaan kertaan. Koska kokeilu oli näin laaja, Maher, Latulipe, Lipford ja Rorrer hyödynsivät toistensa kokemuksia kurssien ”flippaamisessa”.

Kursseilla hyödynnettiin valmista videomateriaalia niin paljon kuin sitä oli saatavilla. Ellei materiaalia ollut, sitä jouduttiin luomaan itse, eli kurssien valmistelutyöstä videoiden teko vei osassa kursseista satoja tunteja. Kursseilla opiskelijoiden valmistelutyöhön ennen luokka-aktiviteetteja kuului muun muassa videoluennot, tekstikirjojen / artikkelien lukeminen ja online-testien teko. Luokka-aktiviteetteja olivat muun muassa prototyyppien teko, testaaminen, arviointi, vertaisarviointi, pseudokoodiongelmien ratkaisu, muut koodaamiseen liittyvät tehtävät ja tunnilla tehtävät testit. Erilaisiin ryhmäaktiviteetteihin / ryhmätöihin kuuluivat muun muassa ad hoc -ryhmätyöt, pariohjelmointi ja koko lukukauden kestävä ryhmätyö. (Maher ym. 2015, 219.)

Käänteisen opetuksen järjestämiseen – miten oppimateriaali annetaan ja milloin harjoitellaan – on useita erilaisia tapoja. Pääsääntöisesti tutkitulla neljällä kurssilla opiskelijoiden työ jakaantui ensin valmistelutyöhön omalla ajalla, ja sen jälkeen luokassa järjestettyihin käytännön harjoitteisiin, jotka olivat ennalta määrättyjä. Kurssien aikana huomattiin, että varsinkin ensimmäisen vuoden opiskelijoita auttaa selkeä struktuuri, jota noudatetaan.

Esimerkiksi Moodlessa hyödynnettiin Moodlen ominaisuutta, jolla pääsyä seuraaviin tehtäviin ei saanut ennen kuin vaaditut valmistelutyöt oli merkattu suoritetuiksi. (Maher ym. 2015, 220.)

Kursseilta kysyttiin palautetta ja vastauksia saatiin 213 opiskelijalta. Palautteessa kysyttiin sekä kysymyksiä, joihin vastattiin likert-asteikolla (1-7), että avoimia kysymyksiä. Opiskelijoiden kursseille antama palaute oli pääsääntöisesti positiivista ja he olivat sekä ymmärtäneet käänteisen opetuksen tarkoituksen, että pitäneet sitä hyödyllisenä oppimisen kannalta. Vastausten perusteella käytetty opetusmetodi paransi oppilaiden sitoutumista sekä oppimateriaaliin, että muihin opiskelijoihin. Opiskelijoilta saatiin myös rakentavaa kritiikkiä, joiden avulla kursseja pystyy kehittämään jatkossa. Palautetta antaneista opiskelijoista vähemmistö kaipasi lisää luokassa luennointia, eikä osa ollut ymmärtänyt, että valmistelutyöt ennen luokassa pidettyjä tunteja kuuluivat myös kurssiaikaan. (Maher ym. 2015, 222-223.) Toivolankin mukaan käänteisessä oppimisessa 85 % oppilaista on kykeneviä itseohjautuvaan toimintaan, kun taas n. 15 % oppilaista ei siihen pysty (Toivola, 2015).

2.4 Arviointi

Kun puhutaan oppimisesta, puhutaan myös sen arvioinnista. Mutta ei ole yhtä ainoaa oikeaa, parasta tapaa arvioida oppimista. Arviointi pohjautuu aina johonkin konkreettiseen, mitattavaan ”todisteeseen”, jonka voi nähdä, jota voi koskea tai jonka voi kuulla. (Griffin & Care 2014, 1-3).

Arviointia voidaan tehdä monella eri tapaa. Erilaisia arviointityyppejä ovat muun muassa diagnostinen, formatiivinen ja summatiivinen arviointi (katso taulukko 1). Näihin kuuluvia menetelmiä on lukuisia, muun muassa jatkuva arviointi, itsearviointi, vertaisarviointi, kurssin lopuksi suoritettava tentti tai muu tehtävä ja sen arviointi. Erilaiset arviointityypit ja -menetelmät ovat hyödyllisiä eri tilanteissa, ja niihin vaikuttavat myös oppimisen tavoitteet.

Taulukko 1 Erilaisia arviointityyppejä

Arviointityyppi	Merkitys
Diagnostinen	Kartoitetaan oppijan oppimisedellytyksiä. Lähtötason määrittely.
Formatiivinen	Opintojen aikana, miten opiskelija on edistynyt suhteessa asetettuihin tavoitteisiin. Jatkuva arviointi.
Summatiivinen	Jälkikäteen, miten asetettuihin tavoitteisiin on päästy. Loppuarviointi.

Formatiivinen arviointi mittaa opintojen aikana opiskelijan edistymistä jatkuvan arvioinnin keinoin. Erilaisia menetelmiä ovat muun muassa vertais- ja itsearviointi sekä reflektointi. Näitä menetelmiä suositellaan käytettäväksi kurssin aikana säännöllisesti, jotta opiskelija saa itsekin tietoa oppimisestaan ja tasostaan (Crisp 2011, 6). Formatiivinen arviointi edesauttaa oppimista ja kertoo sekä opiskelijalle, että opettajalla, mitä oppilaan pitäisi seuraavaksi edistääkseen oppimistaan (Pavlovic ym. 2014, 60).

Summatiivinen arviointi mittaa esimerkiksi loppukokeella, miten opiskelija on suoriutunut. Vertailu voidaan tehdä muihin opiskelijoihin tai opettajan (tai muun instanssin) määrittelemien kriteerien mukaisesti. Summatiivinen arviointi voidaan tehdä numeroilla tai antamalla palaute. Jos arviointi on tehty numeroilla, se ei välttämättä kerro osaamisen tasosta esimerkiksi seuraavalla kouluasteella paljoakaan. Summatiivinen arviointi kertoo tietyn hetken tilanteen, eikä arvioi oppimista prosessina. (Pavlovic ym. 2014, 59-60.) Summatiivisessa arvioinnissa suuri merkitys on loppukokeella, ja sillä, mittaako se luotettavasti kurssilla opetettuja ja opittuja asioita.

Usein formatiivista arviointia käytetään summatiivisen arvioinnin kanssa yhdessä – formatiivisia menetelmiä käytetään kertomaan, mitä opiskelijan pitää vielä tehdä saavuttaakseen summatiivisessa arvioinnista tietyn tason. Arviointia voidaan tehdä myös hyödyntäen pelkästään formatiivisen arvioinnin keinoja, esimerkiksi vertaisarvioinnilla. Tämä aiheuttaa niin sanotun

vapaamatkustajaongelman: pääseekö joku opiskelija kurssista läpi muiden siivellä, kun oppimista ei mitatakaan summatiivisesti.

Oppimisen ja arvioinnin apuna käytetään myös pelillistämisen (gamification) eri keinoja. Opiskelijoille voidaan asettaa tavoitteita, jotka saavutettuaan he saavat jonkun palkinnon, esimerkiksi sähköiseen oppimisjärjestelmään pokaalin. Palkintoina voi myös olla lisäpisteitä kurssipistemäärään, kun joku kurssille asetetuista tavoitteista on minimin yläpuolella. Pelillistämisen edellytyksenä näissä tapauksissa on, että kurssien pistemäärät ovat näkyvillä koko ajan myös opiskelijoille. Pelillistäminen voi auttaa opiskelijoiden motivaatiota kurssin suorittamiseen. (Kaila ym. 2018, 689-690.) Pelillistämisen keinot ovat formatiivisia, ja auttavat opiskelijat näkemään oman edistymisensä kurssin aikana.

3 Verkko-oppimisympäristöt ja sähköinen tentti

3.1 Verkko-oppimisympäristöt

Verkko-oppimisympäristöllä tarkoitetaan oppimisympäristöä, joka on tarkoituksen mukaisesti opiskelijalle pedagogisesti ja teknisesti verkkoon suunniteltu avoin ympäristö. Verkko-oppimisympäristössä on opiskelijan käytössä opiskeluun liittyvä materiaali, tehtävät ja ohjaus. (Viitala & Lehtelä 2009, 51.) Riippuen kuitenkin verkko-oppimisympäristön roolista kurssilla näitä kaikkia ei välttämättä ole käytössä. Verkko-opetusta ei esimerkiksi ole se, että materiaali siirretään sähköiseen muotoon, josta se on helposti saatavilla (Hiltunen, 2012, 37).

Eija Siltarin luennolla verkko-opetuksen suunnittelusta käsiteltiin erilaisia verkko-oppimisympäristöjen rooleja. Verkko-oppimisympäristön rooli muokkaantuu kurssin ja sen tarpeiden mukaisesti. Verkko-oppimisympäristö voi olla osa oppimisympäristöä siten, että se toimii lähiopetuksen materiaalin jakokanavana, tiedotuksessa ja tehtävänannoissa. Monimuoto-opintojen ollessa kyseessä verkko-oppimisympäristö on olennainen osa oppimisympäristöä, ja edellä mainittujen lisäksi siellä on vuorovaikutus opettajan ja oppilaiden välillä, esimerkiksi keskustelualueilla, opiskeluohjeet ja tehtävien palautus ja arviointi. Verkko-opinnoissa verkko-oppimisympäristö on varsinainen oppimisympäristö. Silloin verkossa tapahtuu kaikki. Edellä mainittujen toimintojen lisäksi siellä on yhteistoiminnallinen oppiminen, osaamisen jakaminen ja näkyväksi tekeminen, verkkoluennot. (Siltari, 2018.)

Verkko-oppimisympäristöissä tapahtuvaa vuorovaikutusta on eri tasoilla, eikä pelkästään opiskelijoiden kesken tai opiskelijan ja opettajan välillä. Myös oppimateriaaliin rakennettu automaattinen arviointi on vuorovaikutusta. Linkit ulos verkko-oppimisympäristöstä muuhun verkkomateriaalin missä tahansa verkossa on myös vuorovaikutusta. (Viitala & Lehtelä 2009, 51)

Verkko-opetuksessa ja verkko-oppimisympäristöissä voidaan hyödyntää lukuisia erilaisia pedagogisia malleja riippuen siitä, millainen kurssi on

kyseessä. Valintaan vaikuttaa muun muassa onko opiskelu itseohjautuvaa vai ohjattua, opiskellaanko yksin vai ryhmässä, jaetaanko kurssilla vain tietoa vai tuottaako opiskelija kurssilla jotain, millainen arviointi kurssilla on, ja mitä kaikkea tietoa käytetään arviointiperusteena. Malleja valitessa on muistettava myös muuan muassa oppimista estävät ja edistävät tekijät sekä erilaisten oppimistyylien tukeminen. (Hiltunen, 2012, 43.)

Verkko-oppimisympäristöjä on lukuisia ja niitä on myös eri tarkoituksiin. Esimerkiksi Moodle on verkko-oppimisympäristö, joka on muokattavissa jokaisen oppilaitoksen tarpeiden mukaisesti. Sen tarkoitus on tukea opiskelijoiden oppimista, vuorovaikutusta ja yhteisöllisyyttä kursseilla. Moodlessa on lukuisia erilaisia mahdollisuuksia esimerkiksi materiaalin jakoon ja tehtävien luomiseen. Moodle toimii erityisen hyvin tehtävien palautukseen, palautteen antamiseen ja arviointiin.

Opettajan rooli verkko-oppimisympäristöissä riippuu paitsi valitusta verkko-oppimisympäristö, tavasta käyttää sitä, mutta myös valitusta pedagogisesta mallista (katso luku 2.3). Lisäksi eri ryhmien kesken opiskelijoilla on erilaisia tarpeita. Riitta Suomisen ja Satu Nurmelan (2011, 32) mukaan opiskelijat voidaan jakaa ainakin neljään ryhmään:

1) opiskelijat, jotka tarvitsevat tukea, mutta eivät halua sitä,
2) opiskelijat, jotka tarvitsevat tukea ja haluavat sitä,
3) opiskelijat, jotka eivät tarvitse tukea, mutta haluavat sitä,
4) opiskelijat, jotka eivät tarvitse, eivätkä halua sitä.

Opettajan rooli verkko-oppimisympäristöissä on paljon muutakin kuin materiaalin siirtämistä verkkoon. Opettaja suunnittelee kaiken verkossa tapahtuvan toiminnan, yhteistyön ja vuorovaikutuksen. (Suominen & Nurmela, 2011, 35.) Opettajan rooli on myös toimia keskustelun avaajina, luotsata sitä, antaa palautetta, tukea ja ohjausta. Opettajan roolilla voi olla monta nimeä, opettajan lisäksi esimerkiksi ohjaaja, fasilitaattori, tuutori, valmentaja ja tukija.

Opettajan rooli korostuu opintojen alussa tai vaiheessa, jossa verkko-opetus on uutta. Alkuun opiskelijat tarvitsevat enemmän ohjausta verkko-

oppimisympäristön ja toimintatapojen omaksumiseen, kuin jo pidempään kyseisellä tavalla opiskelleet. Tämän olen huomannut erityisesti aikuisia opettaessani. Heille perinteinen opettajavetoinen opetus on tutumpaa ja heitä pitää kannustaa verkko-opintojen omatoimisuuteen nuorempia enemmän.

3.2 Sähköinen tentti

Tentillä tarkoitetaan tilannetta, jossa opiskelijalle annetaan tehtäviä. Näiden tehtävien avulla arvioidaan, mitataan tai kontrolloidaan, onko hänen osaamistasonsa riittävä suhteessa asetettuihin tavoitteisiin. Tentti pidetään kurssin lopussa, ja siitä on päästävä läpi, jotta voi edetä koulutuksessa. (Karjalainen, 2001, 12.)

Perinteinen tentti on kurssin keskellä tai päätteeksi pidettävä kirjallinen tentti. Tentti suoritetaan erityisessä, ennalta määrätyssä tenttisalissa tai luokkahuoneessa. Tenttijä vastaa tehtäviin itsenäisesti, valvottuna, rajallisen ajan puitteissa ja muistinvaraisesti. (Karjalainen, 2001, 13.)

Sähköisellä tentillä tarkoitetaan tenttiä, joka tehdään tietokoneella. Tentti voidaan tehdä kotona tai ennalta määrätyssä tilassa, omalla (kannettavalla) tietokoneella tai oppilaitoksen antamalla tietokoneella, valvottuna tiettyyn aikaan, ilman valvontaa tiettynä aikana, yksin tai ryhmässä. Mahdollisuuksia sähköisen tentin järjestämiseen on useita. Rajoituksia tentille voi luoda esimerkiksi valitun tenttijärjestelmän tehtävätyyppien rajoitukset.

Turun yliopiston Tulevaisuuden teknologioiden laitoksella erilaisia sähköisen tentin järjestelmiä ovat ViLLE, Moodle ja Exam. Eri järjestelmillä on omat etunsa ja haittansa, ja järjestelmä onkin valittava käyttötarkoituksen mukaan. ViLLEstä lisää luvussa 4.

Moodlessa on mahdollista järjestää myös kokeita tai tenttejä. Yksinkertaisimmillaan Moodlea voidaan pitää valvotun tietokoneella tehtävän kokeen palautuslaatikkona, jolloin palautusaika on rajattu vain tentin ajaksi.

Moodleen voidaan myös luoda erilaisia kokeita, jotka voivat sisältää esimerkiksi monivalintatehtäviä, aukkotehtäviä ja esseetyyppisiä tehtäviä. Kokeiden kysymykset määritellään Moodlessa, ja esimerkiksi monivalintatehtäviin opettaja kirjaa järjestelmään myös oikeat vastaukset, eli Moodle korjaa ne myös automaattisesti. Vastausvaihtoehdot voidaan esittää jokaiselle opiskelijalle satunnaisessa järjestyksessä. Esseetyyppisissä tehtävissä opettaja korjaa tehtävät manuaalisesti ja lisää pisteet Moodleen.

Esseetyyppinen vastauslaatikko antaa myös mahdollisuuden tehdä ohjelmoinnin tehtäviä Moodle-tenteissä, mutta silloin käytössä ei ole ohjelmoinnin työkaluja. Opettaja voi nämä sallia tenttitilanteessa käyttämällä Moodlen ulkopuolisia ohjelmia.

Moodlen tentit voidaan järjestää valvottuina tai etätentteinä. Etätentteinä voidaan järjestää esimerkiksi esseetyyppisiä kokeita, joissa kurssimateriaalia saa käyttää esseen tekemisessä vapaasti. Silloinkin kyseessä on vain tietty, määrätty aika, jolloin tentti on auki, ja vastauksen palautettua sitä ei voi enää muuttaa. Moodle mahdollistaa siten myös harjoitustenttien tekemisen opiskelijoille. Harjoitustentistä lisää seuraavassa alaluvussa.

Exam on Turun yliopiston sähköinen tenttijärjestelmä, joka on otettu käyttöön vuoden 2018 aikana. Tentti on videovalvottu, eikä tentissä saa käyttää mitään oheismateriaalia. Sähköisessä tenttijärjestelmässä voi toteuttaa tenttien lisäksi myös kypsyysnäytteitä. Tenteistä Examiin sopivat parhaiten sellaiset, joiden kysymyksiä järjestelmä voi satunnaistaa jokaiselle opiskelijalle erilaiseksi opettajan muokkaamasta kysymyspankista. Erilaisia tehtävätyyppejä on kolme: essee, monivalintatehtävät (yksi ja monta oikein) ja aukkotehtävät. Examissa järjestettyihin tentteihin voi liittää mukaan liitteitä pdf-, doc- ja kuvamuodossa. (Turun yliopisto, 2019.)

Examin edeltäjässä Tenttiksessä oli rajoitteena erilaisten tenttien tekemiseen se, että vastaus oli pelkkää leipätekstiä, eikä opiskelija voinut käyttää mitään apuohjelmia. Examissa opiskelijan käytössä on lukuisia apuohjelmia, joilla opiskelija voi kirjoittaa vastauksensa, ja sen jälkeen hän voi liittää

tallentamansa tiedoston vastaukseen. Käytettäviä ohjelmia ovat Word (ilman oikolukua), Excel, Paint, PowerPoint, Dia, Inkscape, RStudio, Note, pdf-luku ja tallennusohjelma ja peruslaskin. Matemaattisia kaavoja varten opiskelijalla on käytössä TeX. (Turun yliopisto, 2019.)

Vaikka Exam on huomattavasti kattavampi kuin aiempi sähköisen tentin järjestelmä, ei Examilla edelleenkään voi käyttää ohjelmoinnissa oleellisia työkaluja, kuten esimerkiksi debuggaus ja validointi.

3.3 Harjoitustentti

Harjoitustentillä tarkoitetaan tässä yhteydessä tenttiä, joka järjestetään ennen varsinaista tenttiä samassa ympäristössä samantapaisilla tenttikysymyksillä kuin varsinainen tentti. Harjoitustenttiä ei arvioida, eikä se välttämättä ole yhtä laaja kuin varsinainen tentti. Harjoitustentillä voi harjoitella tentissä olevia tehtävätyyppejä tai tenttiympäristön käyttöä. Esimerkiksi sähköisen tentin järjestelmä voi olla sellainen, että se vaatii – ainakin joillekin opiskelijoille – harjoituskerran ennen tentin suorittamista. Optimaalisesti tenttiä ei suoritettaisi sellaisessa sähköisessä ympäristössä, joka ei ole opiskelijalle tuttu.

Harjoitustentin muoto on yleensä sama kuin varsinaisen tentin ja siinä olevat kysymykset ovat samantapaisia kuin loppukokeessa. Harjoitustentissä saa pitää materiaalia esillä, eikä kukaan valvo sen suorittamista. Sähköisen harjoitustentin etuja on palaute eli opiskelija saa harjoitustentin jälkeen tietää, miten suoriutui tentistä. (Gehrig 2017.) Kurssikohtaisesti harjoitustentin voi suorittaa joko kerran tai useamman kerran.

Harjoitustenttien vaikutusta nimenomaan ohjelmoinnissa ei ole tutkittu paljon. Sen sijaan muissa yliopistotason kursseissa niitä on tutkittu. Renee Oliver ja Robert L. Williams (2005, 144-146) tutkivat harjoitustentin vaikutusta loppuarvosanaan ihmisen kehitykseen liittyvällä kurssilla. Kurssi koostui viidestä osa-alueesta, joista jokaiseen kuului välikoe. Välikoetta ennen opiskelijat tekivät harjoituskokeen, joka palautettiin ja tarkistettiin yhdessä

luennolla. Tällä tavalla opiskelijat näkivät heti, miten olivat suoriutuneet kokeessa. Harjoituskokeesta sai myös pisteitä kurssiarvosanaan, maksimissaan 5 % kurssin kokonaispistemäärästä. Osa ryhmästä sai pisteet palauttamalla harjoituskokeen ja osa ryhmästä sai pisteet harjoituskokeen oikeiden vastausten perusteella.

Tutkimuksen tuloksissa huomattiin, että harjoituskokeista oli hyötyä opiskelijoiden loppuarvosanaan, erityisesti niille opiskelijoille, jotka saivat pisteitä harjoituskokeista sen perusteella, olivatko vastaukset oikein. Tutkimusta ei kuitenkaan tehty laboratorio-olosuhteissa, joten kaikkia muuttujia ei voitu kontrolloida, jotka olisivat voineet vaikuttaa opiskelijoiden tuloksiin. Esimerkiksi tutkijoilla ei ole tietoa, tekivätkö opiskelijat harjoituskokeet yksin vai ryhmässä tai kopioiko joku toisen vastaukset, koska harjoituskokeen tekemistä ei kontrolloitu (niin sanottu vapaamatkustajaongelma). Opiskelijoille ei myöskään tehty alkutestejä, eli ei ole tietoa, kuinka paljon opiskelijoiden tulokset paranivat tilanteesta ennen kurssia. Lisäksi eri osioita opettivat eri opettajat (teacher assistants), eli vaihtelua tuli jo opetuksen tasossa. (Oliver & Williams 2005, 148-150.)

Tutkimuksen päätelmä oli, että harjoituskokeista on hyötyä kurssilla, jos ne vastaavat tyyliltään / muodoltaan loppukoetta, ja ovat myös vaikeustasoltaan samantasoisia kuin loppukokeet. Harjoituskokeiden hyöty loppukokeen arvosanaan on suurempi, jos harjoituskokeista saatavat mahdolliset pisteet annetaan oikeiden vastausten perusteella. Opiskelijat näkevät silloin vaivaa harjoituskokeen tekoon, ja oikeasti miettivät vastauksiaan. Jos harjoituskokeesta saa pisteet vain palauttamalla, osa opiskelijoista ei näe vaivaa niiden tekemiseen ja se voi vaikuttaa myös heikentävästi loppukokeen arvosanaan. (Oliver & Williams 2005, 151.)

Peter Reuterin ja Stacey Parkerin (2015) konferenssiesityksessä esiteltiin harjoituskokeen vaikutuksia loppuarvosanaan Ihmisen anatomia ja fysiologia I -kurssilla. Harjoituskokeet tehtiin verkko-oppimisympäristössä ja ne vastasivat muodoltaan loppukokeita eli niissä oli sama määrä samantyyppisiä kysymyksiä kuin loppukokeessa. Opiskelijoille annettiin myös harjoituskokeen

suorittamiseen sama aika kuin loppukokeeseen. Tuloksista huomattiin, että opiskelijoiden, jotka tekivät kaikki neljä harjoituskoetta, läpäisyprosentti kurssista (80,95 %) oli huomattavasti korkeampi kuin yleisesti kurssilla 65,7 %). Opiskelijoiden, jotka eivät tehneet harjoituskokeita lainkaan, läpäisyprosentti kurssilla oli 50,0 %. Tutkimuksessa huomattiin myös, että harjoituskokeiden suorittaminen ei vaikuttanut korkeimman arvosanan saaneiden prosentuaaliseen määrään, vaan se pysyi suunnilleen samana riippumatta harjoituskokeiden suorittamisesta.

Molemmissa case-tapauksissa harjoitustentistä oli hyötyä loppukokeen suorittamiseen, kun ne vastasivat muodoltaan loppukoetta. Lisäksi harjoituskokeiden arvioinnilla on merkitystä. Jos harjoituskokeesta saa arvosanan tai esimerkiksi pisteitä, jotka vaikuttavat loppuarvosanaan, niihin panostetaan enemmän, joten opiskelijat myös opiskelevat enemmän.

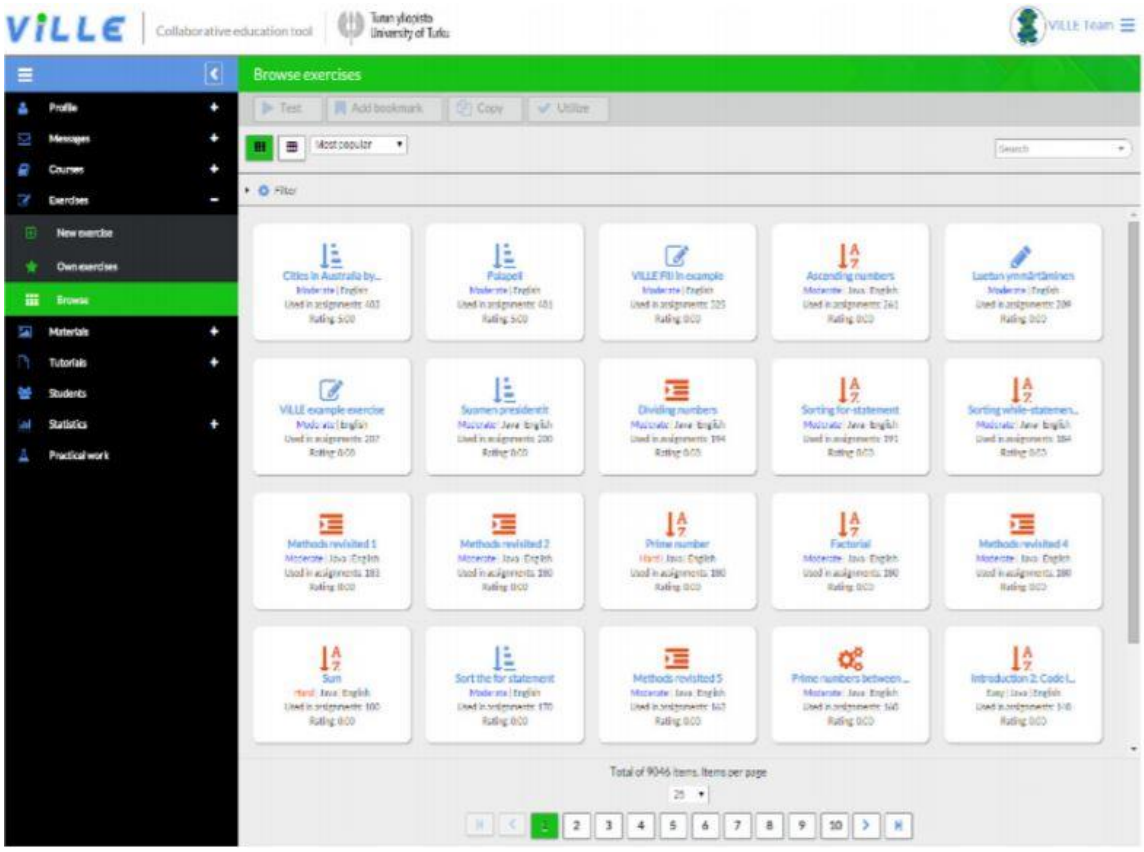
4 ViLLE

ViLLE on Turun yliopiston Tulevaisuuden teknologioiden laitoksella kehitetty oppimisjärjestelmä, joka toimii verkossa. Alun perin se kehitettiin visualisoimaan ohjelmia ja algoritmeja, koska visualisoinnin todettiin auttavan muuten niin abstraktin ohjelmoinnin oppimista. Se kehitettiin erityisesti matematiikan ja ohjelmoinnin opetusta varten, mutta sisältää nykyisin lukuisia erilaisia tehtävätyyppejä, joita voidaan hyödyntää melkein minkä tahansa aineen kurssilla. ViLLEssä on myös automaattisesti arvioituja tehtäviä, joista opiskelija saa välitöntä palautetta. Tästä on erityisesti apua oppimiseen. (Laakso, Kaila & Rajala 2018, 1656-1662.)

4.1 Järjestelmän kuvaus

ViLLen viimeisin versio on kehitetty käyttäen kokemuksia ViLLEn aiemmista versioista. ViLLEn käyttämiseen ei tarvitse erillistä ohjelmaa, vaan ViLLE toimii tavallisessa selaimessa, eikä sen käyttöön tarvita mitään lisä-plugineja. Kaikki tiedon prosessointi tapahtuu palvelimella. ViLLEn viimeisimpään versioon on luotu lukuisia erilaisia tehtävätyyppejä, ja uusien tehtävätyyppien luomiseksi on luotu käyttöliittymä, jotta tehtävätyyppien lisääminen jatkossakin on helppoa. ViLLE käyttää Java-sovelluskehys Vaadinta. (Laakso, Kaila & Rajala 2018, 1657-1658.)

ViLLEn suunnittelussa on ollut neljä kulmakiveä: opettajien yhteistyö, opiskelijoiden yhteistyö, automaattinen arviointi ja välitön palaute. Opettajien yhteistyötä helpottaa se, että kaikki ViLLEen luotavat tehtävät, kurssit ja muu materiaali ovat automaattisesti kaikkien ViLLEn (opettaja)käyttäjien käytössä (katso seuraavalta sivulta kuva 6, ViLLEn opettajanäkymä ja julkisten tehtävien listaus). Näistä kaikista saa myös yksityisiä määrittelemällä sen kyseisen resurssin asetuksista. (Laakso, Kaila & Rajala 2018, 1657.)



Kuva 6 ViLLE Opettajan näkymä - Selaa julkisia tehtäviä.

ViLLEssä perusyksikkö on kurssi. Kurssit on jaettu kierroksiin, jotka voivat sisältää erilaisia tehtäviä. Kierrokset voivat sisältää esimerkiksi yhden viikon tehtävät, kurssiläsnäolot tai johonkin tiettyyn aihepiiriin liittyvät tehtävät. Kierroksia voi myös olla eri tyyppisiä, muun muassa tenttikierroksia, tutoriaalikierroksia tai ehdollisia kierroksia. Näillä kaikilla on omia toisistaan poikkeavia ominaisuuksia. Esimerkiksi ehdolliset kierrokset avautuvat vasta, kun sille asetetut ehdot täyttyvät. (Laakso, Kaila & Rajala 2018, 1658.)

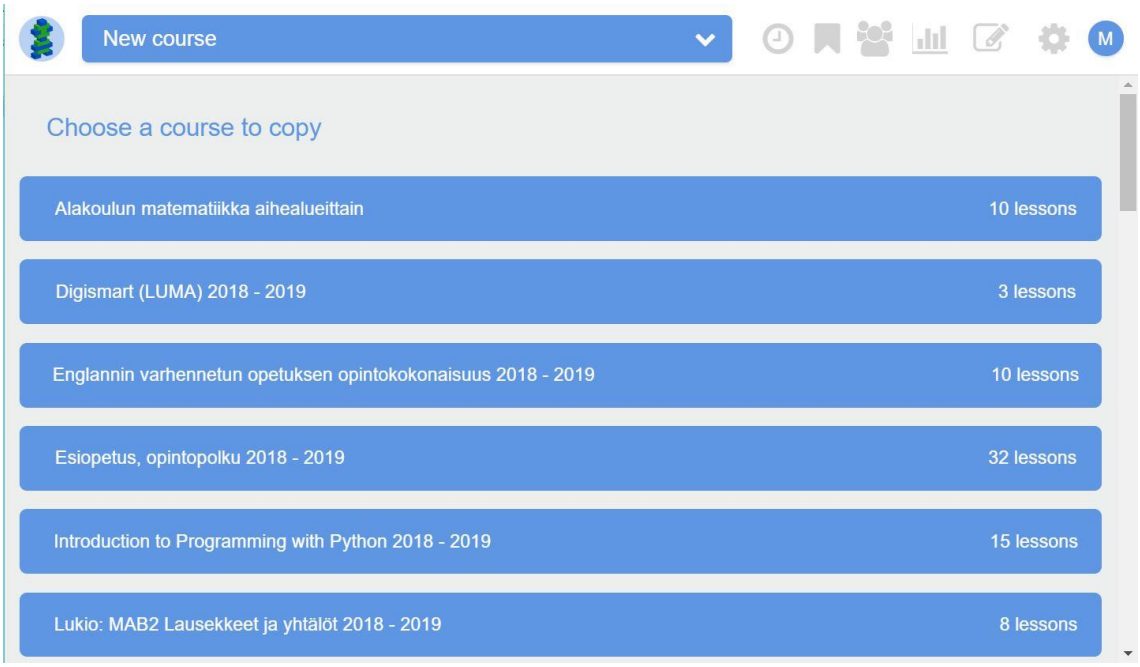
ViLLEssä on lukuisia erilaisia tehtävätyyppejä. ViLLEn ohjelmointitehtävätyypeillä opiskelijat voivat kirjoittaa ja suorittaa koodia useilla ohjelmointikielillä. Lisäksi erilaisten tehtävien avulla voi harjoitella algoritmeja tai nähdä ohjelmoimansa ohjelman visualisoituna. Tehtäviin saadaan myös automaattinen välitön palaute tarvittaessa. ViLLEssä on kattava oppimisanalytiikka taustalla, ja opettaja saa halutessaan paljon dataa opiskelijoiden tehtävien suorittamisesta. (Oppimisanalytiikan keskus, 2018.)

Tutoriaalit ovat yhdistelmä erilaisia tehtäviä ja oppimateriaaleja. Tehtävät ovat yleensä automaattisesti arvioituja ja oppimateriaali voi sisältää muun muassa videoita ja luentomonisteita. Tutoriaaleja tehdään yleensä pareittain tai pienryhmissä. ViLLEssä olevat tutoriaalit pohjautuvat konstruktivistiseen oppimiskäsitykseen. Tarkoituksena on tehdä opiskelijoista aktiivisia oppijoita. Opettajan rooliksi jää olla enemmänkin ohjaaja. (Oppimisanalytiikan keskus, 2018.) Tutoriaaleissa ilmenee hyvin ViLLEn suunnittelusta kolme kulmakiveä eli opiskelijoiden yhteistyö, automaattinen arviointi ja välitön palaute

ViLLEä voidaan hyödyntää myös luentokäytössä, esimerkiksi keräämällä läsnäoloja RFID:n avulla. Tämän avulla opettajat näkevät reaaliaikaisesti, ketkä ovat paikalla ja läsnäoloista voidaan helposti antaa pisteitä. Läsnäolotietoja voidaan myös hyödyntää esimerkiksi demonstraatioiden yhteydessä jakamalla demovuoroja automaattisesti läsnäolijoiden kesken.

Kuten jo mainitsin, ViLLEssä on taustalla kattava oppimisanalytiikka ja dataa kerätään paljon opiskelijoiden suorituksista automaattisesti. Tietoa kerätään muun muassa palautuksista, kuinka kauan tehtävien tekemiseen menee aikaa, kuinka monta kertaa opiskelija palautti saman tehtävän, mihin aikaan tehtävä on palautettu. Lisäksi jossain tehtävätyypeissä tietoa kerätään yksittäisten opiskelijan hiiren klikkauksista lähtien. Kaikki data on nähtävissä ViLLEn tilastonäkymässä. Se voidaan myös viedä Exceliin tai muuhun taulukkolaskentaohjelmaan tai CSV-tiedostoon. (Laakso, Kaila & Rajala 2018, 1664.)

ViLLEn opettajanäkymän saa myös yksinkertaistettuna, kevyempänä versiona. ViLLE Lite sisältää ViLLEn eniten käytetyt työkalut kurssien hallintaan. Kevyemmässä näkymässä voi myös luoda uusia kursseja kopioimalla olemassa olevia (kuva 7).



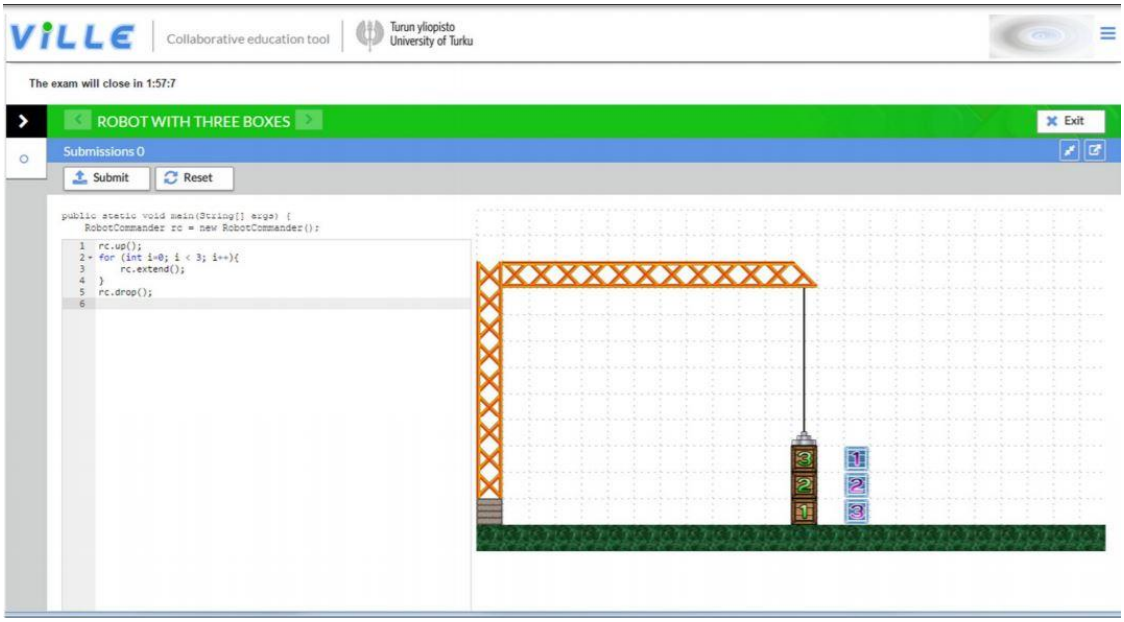
Kuva 7 ViLLE Lite -näkymässä olemassa olevan kurssin kopiointi omaan käyttöön.

4.2 Sähköinen tentti ViLLEssä

ViLLEssä tentti voi sisältää sekä automaattisesti arvioituja tehtäviä, että opettajan arvioimia tehtäviä. Tehtävätyyppeinä voidaan tentissä käyttää kaikkia ViLLEn tehtävätyyppejä, muun muassa ohjelmointiin erityisesti kehitettyjä tehtävätyyppejä. ViLLE-tentin voi suorittaa myös kotona, jos tentti on määritetty niin, mutta käytännössä tutkituilla ohjelmointikursseilla ViLLE-tentti suoritetaan valvottuna. (Rajala ym. 2016.)

ViLLEssä sähköinen tentti on rakenteeltaan samanlainen kuin kurssin aikana tehdyt ViLLE-tehtävät. Tentissä voi käyttää kaikkia samoja elementtejä kuin harjoitustehtävienkin rakentamiseen. Rakennettaessa tenttiä valitsemalla kierroksen tyypiksi ”tenttikierros” voidaan kuitenkin halutessa piilottaa välitön palaute ja pistemäärät tenttikierroksilta (Kaila & Kurvinen, 2018). Samoin tentistä voidaan piilottaa osa tai kaikki käytössä olevista avusteista. Usein tämä on perusteltua tentin aikana, koska halutaan testata opiskelijan oppimaa tietoa tähän mennessä.

Tentissä tehtävien yrityskertoja ei ole rajoitettu. Tentillä on kuitenkin aikaraja, jonka sisällä tentti on suoritettava. Ohjelmointitenteissä opiskelija saa kuitenkin tietoonsa kääntäjän antamat virheilmoitukset ja ajonaikaiset virheet. Tämä auttaa opiskelijaa keskittymään koodin rakenteelliseen oikeellisuuteen, eikä murehtimaan syntaksivirheistä. Mitään muuta palautetta tentissä ei tule, jos välitön palaute on otettu pois käytöstä. (Rajala ym. 2016.)



Kuva 8 Robottitehtävä tentissä ViLLEn opiskelijanäkymässä.

Ohjelmointitenttien tehtävätyyppejä voivat muun muassa olla kysely, koodaustehtävä, robottitehtävä, koodin järjestystehtävä ja yhdistämistehtävä. Kyselyssä voi olla monivalintakysymyksiä tai lyhyitä avoimia vastauksia. Koodaustehtävässä opiskelija kirjoittaa ohjelman tai sen osan annetun tehtävän mukaisesti. Robottitehtävässä (katso kuva 8) opiskelija kirjoittaa ohjelman, jolla robotti liikkuu / toimii annetun ohjeen mukaisesti. Koodin järjestystehtävässä on useita pieniä paloja koodia, jotka pitää järjestää oikeaan järjestykseen. Yhdistämistehtävässä yhdistetään kaksi toisiinsa liittyvää asiaa toisiinsa. (Rajala ym. 2016.)

ViLLEn sähköinen tentti tallentaa kaiken datan palvelimille, jotka ovat saatavilla mistä vain, milloin vain. Käytettäessä vain automaattisesti arvioitavia

tehtäviä, tulokset ovat valmiit heti kun opiskelija palauttaa kokeen. (Laakso, Kaila & Rajala 2018, 1663.) ViLLEn automaattisesti arvioitavat tehtävät ovat yleensä perusteltuja isoilla kursseilla, jolloin opettajan manuaalisesti arvioimat tehtävät veisivät enemmän aikaa kuin resursseja on käytössä.

Sähköinen tentti hyödyttää myös opettajan arvioimissa tehtävissä. Opiskelija pystyy muokkaamaan vastauksiaan helposti, ja jäsentämään sitä eri tavalla kuin paperille kirjoittaessaan. Lisäksi tietokoneella kirjoitettua tekstiä on helpompi lukea.

5 Tutkimuksen tarkoitus ja tutkimusmenetelmä

5.1 Tutkimuksen tarkoitus ja tutkimuskysymykset

Tutkimuksen tarkoitus on selvittää, miten Turun yliopiston ohjelmoinnin peruskursseilla vapaaehtoisena suoritettava harjoitustentti vaikuttaa kurssin lopputentin pistemäärään.

Tutkimuskysymykset ovat:

1. Vaikuttaako ViLLE-oppimisympäristössä tehty vapaaehtoinen harjoitustentti ohjelmoinnin peruskurssin lopputentin pistemäärään?
2. Miten harjoitustentin vaikutus eroaa tutkittavien ryhmien välillä (IT-pääaineopiskelijat, sivuaineopiskelijat, avoimen yliopiston opiskelijat)?

5.2 Tutkimuksen kohteen / aineiston ja menetelmän esittely

5.2.1 Tutkimuksen kohde ja käytössä oleva aineisto

Tutkittavia kursseja on kolme: Algoritmien ja ohjelmoinnin peruskurssi, Avoimen yliopiston Algoritmien ja ohjelmoinnin peruskurssi ja Introduction to Programming. Kaikkien tutkittavien kurssien sisältö on pääsääntöisesti samanlainen, mutta joitakin eroavaisuuksia on (katso taulukko 2). Niiden kohderyhmät ovat kuitenkin hieman erilaiset. Lisäksi yksi kursseista opetettiin englanniksi. Kaikki kurssit järjestettiin syksyn 2017 aikana Turun yliopistossa Tulevaisuuden teknologioiden laitoksen opettamana. Sain aineiston ViLLE Teamilta / Oppimisanalytiikan tiimiltä anonymisoituna. Osallistuin kesällä 2017 suunnitteluun, mitä tietoja tutkimuksen kohteena olevilta kolmelta kurssilta kerätään.

Taulukko 2 Tutkittavien kurssien perustietoja.

	Algoritmien ja ohjelmoinnin peruskurssi (AOP)	Algoritmien ja ohjelmoinnin peruskurssi (AvoinAOP)	Introduction to Programming (ItP)
Ensisijainen kohderyhmä	Tietojenkäsittelytieteen pääaineopiskelijat, muut, joille kurssi pakollinen	Avoimen yliopiston kautta kaikille	Bioinformatiikan opiskelijat, sivuaineopiskelijat, vaihto-opiskelijat
Opintopisteet	5-6	6	5-6
Kurssin alussa mukana	272	74	96
Maksimipistemäärä kurssilla	1560	1050	944
Ohjelmointikieli	Java	Java	Python
Opetuskieli	suomi	suomi	englanti

Algoritmien ja ohjelmoinnin peruskurssin (jatkossa AOP) osaamistavoitteita ovat olio-ohjelmointikielen peruskäsitteisiin ja rakenteisiin tutustuminen, ohjelmoinnissa tarvittavan algoritmisen ajattelun opettelu ja editorin ja kääntäjän kanssa työskentelyyn riittävä käytännön ohjelmointitaidon hankkiminen. Opinnot on suunnattu ensisijaisesti tietojenkäsittelytieteiden pääaineopiskelijoille ja muille matematiikan ja tilastotieteiden laitosten oppiaineille, joissa kurssi on pakollinen. Kurssille voi osallistua myös sivuaineopiskelijat mistä tahansa tiedekunnasta. Kurssin ohjelmointikielenä käytetään Javaa. Kurssin opintopistemäärä on 5-6. Yhden pisteen voi suorittaa tekemällä harjoitustyön, joka on osalle osallistujista vapaaehtoinen ja toisille pakollinen riippuen pääaineesta ja opintojen aloitusvuodesta. Kurssin opetuskieli on suomi. (Turun yliopisto, 2017c.)

Avoimen yliopiston Algoritmien ja ohjelmoinnin peruskurssi (jatkossa AvoinAOP) on sisällöltään samanlainen kuin AOP, mutta kohderyhmänä on kuka tahansa avoimen yliopiston kautta kurssille osallistuva henkilö. Kurssille osallistuvalta vaaditaan esitietoina joko kurssin ”Tietojenkäsittelyn perusteet I” tai vastaavat tiedot. Lisäksi AvoinAOP:in opintopistemäärä on 6 eli kurssi sisältää aina harjoitustyön. (Turun yliopisto, 2017a.)

Introduction to Programming (jatkossa ItP) oppimistavoitteina on perustaidot algoritmien luomisessa ja yksinkertaisten, käytännönläheisten ohjelmien kirjoittaminen Python-ohjelmointikielellä. Kurssi on suunnattu opiskelijoille, joilla ei ole aiempaa ohjelmointikokemusta. Opiskelijoista suurin osa on bioinformatiikan opiskelijoita ja vaihto-opiskelijoita. Kurssin opetuskieli on englanti. Opintopistemäärä on 5-6, riippuen taas siitä, kuuluuko harjoitustyö pakollisena vai vapaaehtoisena opintoihin. (Turun yliopisto, 2017b.)

Kaikilla kolmella kurssilla on sama perusrakenne. Kurssi koostuu luennoista, tutoriaaleista, demonstraatioista, ViLLE-tehtävistä ja itsenäisestä työskentelystä. Lopullinen arvosana muodostuu läsnäolosta, tehtävistä saaduista pisteistä (tutoriaalit, demonstraatiot, ViLLE-tehtävät) ja tentistä. Lisäksi erilaisista kyselyistä ja palautteesta saa yksittäisiä pisteitä. Kurssin maksimipistemäärä vaihtelee kolmen eri kurssin välillä.

Aineistona minulla on käytössä kolmesta kurssista kaikkien kurssille osallistuneiden pistemäärät jaoteltuna eri tehtäviin. Pisteet on kaikki kirjattu ViLLE-järjestelmään, josta data on otettu xlsx-tiedostoon, jokainen kurssi omaan tauluunsa. Näistä pisteistä näen muun muassa yksittäisten tutoriaalien ja demonstraatioiden pisteet, vapaaehtoisen harjoitustentin pisteet sekä tenttipisteet (kaikilta kolmelta tenttikerralta). Kursseille osallistui yhteensä 442 opiskelijaa (AOP 272, AvoinAOP 74, ItP 96).

5.2.2 Menetelmät

Tutkimukseni on kvantitatiivinen. Käytän olemassa olevaa materiaalia, jonka Oppimisanalytiikan keskus / ViLLE TEAM on kerännyt syksyn 2017 aikana Turun yliopistossa järjestetyiltä kolmelta ohjelmoinnin kurssilta. Datan olen saanut anonymisoituna eli en voi linkittää dataa yksilöihin, vaan sitä käsitellään anonyymina joukkona. Kesällä 2017 osallistuin suunnitteluun, mitä dataa kerätään näiltä kolmelta kurssilta.

Tutkimukseni on luonteeltaan selittävä. Selittävä tutkimus etsii selitystä tilanteelle tai ongelmalle. (Hirsjärvi ym., 2009, 138-139.) Valitsin strategiaksi kvantitatiivisen tutkimuksen. Kvantitatiivisessa tutkimuksessa on keskeistä esimerkiksi esittää hypoteeseja, tehdä johtopäätöksiä aiemmista tutkimuksista, määritellä käsitteitä, suunnitella aineiston keruuta siten, että aineisto sopii määrälliseen, numeeriseen mittaamiseen, kerätä tietoa taulukkomuotoon ja tutkia aineistoa tilastollisen analysoinnin avulla (Hirsjärvi ym., 2009, 140).

Käsiteltävän datan olen saanut xlsx-muotoisena, kolmessa eri taulussa. Dataa on yhteensä 444 riviä, joissa on keskimäärin 35 saraketta.

5.3 Aiheen rajaus ja tutkimuksen kulku

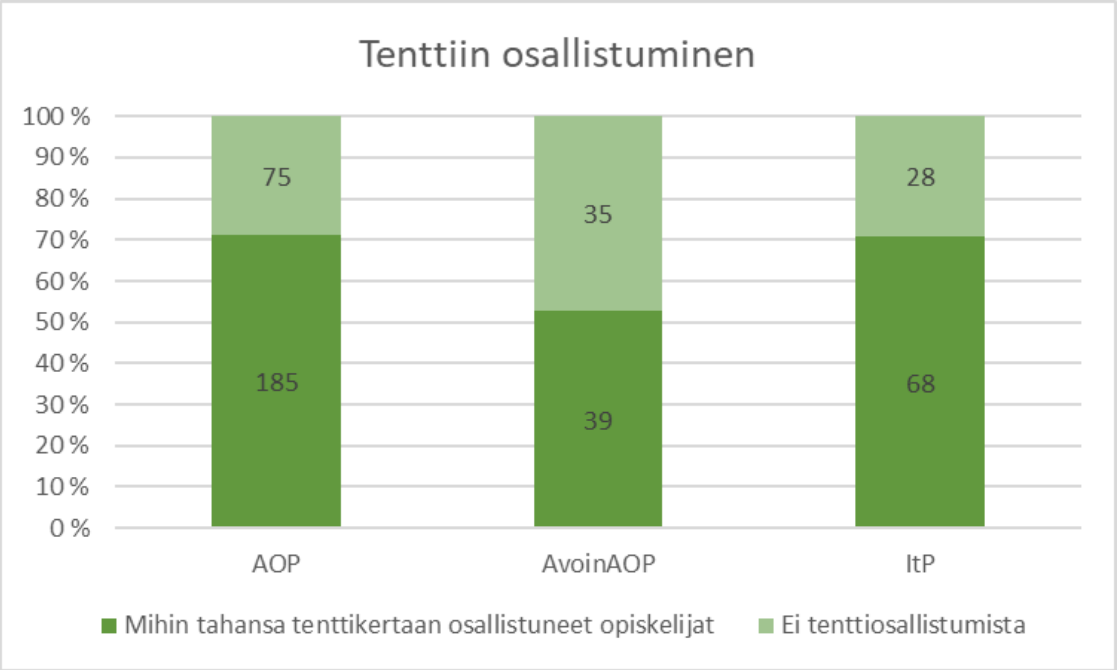
Tutkimukseni aihe on harjoitustentti ja miten se vaikuttaa lopputentin pistemäärään. Tästä syystä alkuvaiheessa rajattiin tutkimuksesta pois muut kursseihin kuuluvat tehtävät ja niiden yksittäiset pistemäärät. Tästä poikkeuksena on kurssin kokonaispistemäärä, jota tutkitaan harjoitustentin ja lopputentin rinnalla. Aluksi tutkittiin vain, onko suorittanut harjoitustentin (kyllä / ei), ja tämän perusteella jaettiin opiskelijat kahteen ryhmään. Tutkimuksessa käytetyt lyhenteet on selitetty taulukossa 3.

Taulukko 3 Selite käytetyistä lyhenteistä.

	Harjoitustentti tehty	Ei harjoitustenttiä
Algoritmien ja ohjelmoinnin peruskurssi	AOP-HT	AOP-EiHT
Avoimen yliopiston Algoritmien ja ohjelmoinnin peruskurssi	AvoinAOP-HT	AvoinAOP-EiHT
Introduction to Programming	ItP-HT	ItP-EiHT

Kuvassa 9 on kuvattu kaikkien kolmen kurssin osalta ViLLEn kurssialueelle kirjautuneiden osallistuminen mihin tahansa kurssin kolmesta tenttikerrasta.

AOP-kurssin osalta tenttiin osallistumisprosentti oli 71,2 %, AvoinAOP:in osalta 52,7 % ja ItP:n osalta 70,8 %.



Kuva 9 Kurssin ViLLE-alustalle kirjautuneiden osallistuminen mihin tahansa tenttikertaan.

Tämän jälkeen aineistosta on rajattu pois ne opiskelijat, jotka eivät suorittaneet tenttiä lainkaan (taulukko 4). Tähän voi olla lukuisia syitä, mutta esimerkiksi vaihto-opiskelijoiden tapauksessa on mahdollista, että tentti järjestettiin heidän jo lähdettyä kotiyliopistoonsa.

Taulukko 4 Aineiston rajaus.

	Algoritmien ja ohjelmoinnin peruskurssi (AOP)	Algoritmien ja ohjelmoinnin peruskurssi (AvoinAOP)	Introduction to Programming (ItP)
Kurssin alussa mukana	272	74	96
Korvaava koe suoritettu hyväksyttävästi	12	-	-
Ei tenttiosallistumista	75	35	28
Ei osallistumista kurssin aikana, vain tentti	2	0	0
N	**183**	**39**	**68**

Lisäksi rajasin tutkimuksen ulkopuolelle opiskelijat, jotka suorittivat tentin, mutta ei muuta kurssiin kuuluvia tehtäviä. Tällaiset opiskelijat ovat mahdollisesti suorittaneet muut kurssiin kuuluvat tehtävät jo aiempina vuosina. Aineistosta rajattiin pois myös opiskelijat, jotka suorittivat kurssin alussa vapauttavan kokeen hyväksyttävästi. Vapauttava koe oli käytössä vain kurssilla AOP ja sen suoritti hyväksyttävästi 12 opiskelijaa.

Rajausten jälkeen tutkittavan datan N eri kursseilla on AOP 183, AvoinAOP 39 ja ItP 68. Yhteensä tutkittavia opiskelijoita on 290.

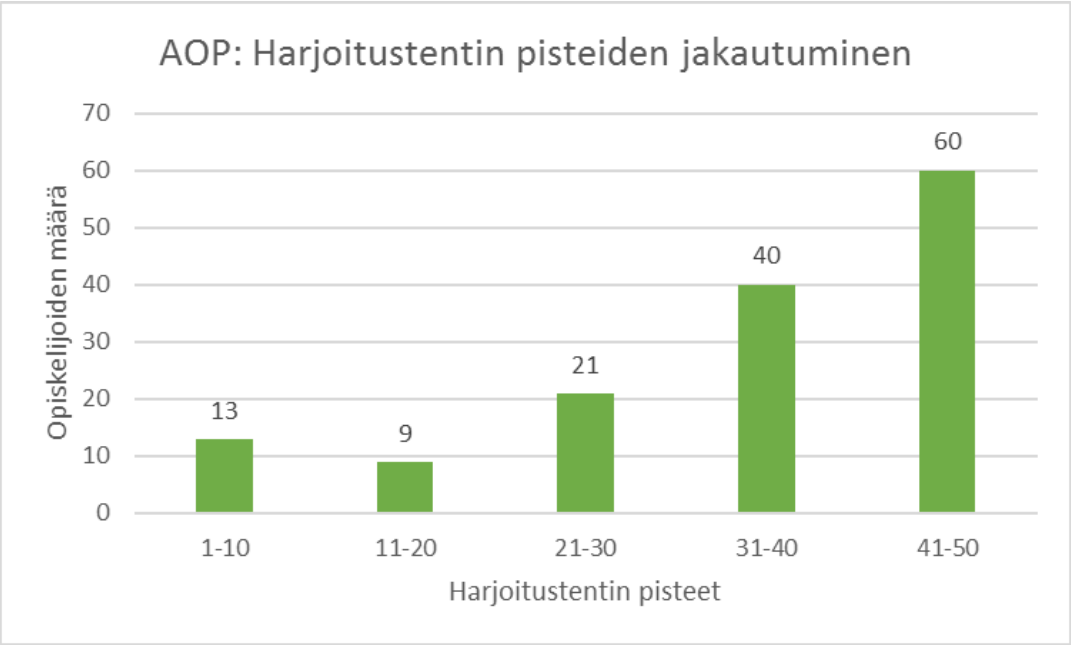
6 Tulokset

Aluksi tutkin eri kurssien harjoitustenttiin osallistumista. Osallistumisprosentit (taulukko 5) vaihtelevat eri kurssien välillä. ItP-kurssin harjoitustenttiin osallistui 88,2 % lopputenttiin osallistujista, kun taas AOP-kurssin harjoitustenttiin osallistui 10 prosenttiyksikköä vähemmän opiskelijoita. AvoinAOP-kurssilla lähes 85% tenttiin osallistuneista teki myös harjoitustentin. Alla on listattu myös harjoitustenttiin osallistujien ja ei-osallistujien lukumäärät, koska kursseilla on paljon eroa osallistujamäärissä.

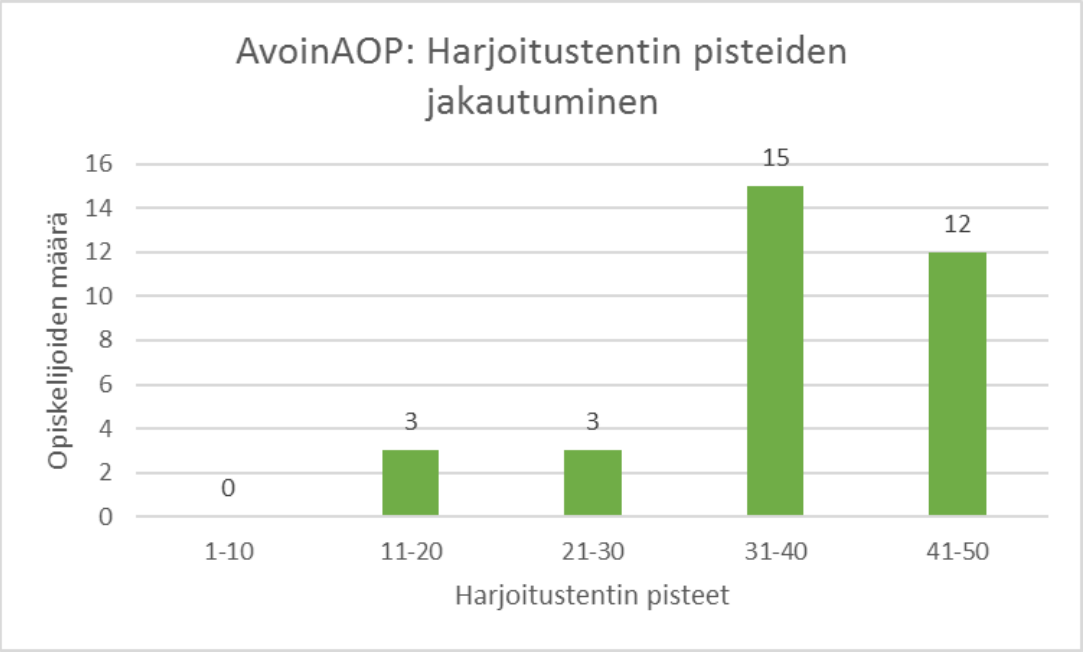
Taulukko 5 Harjoitustenttiin osallistuminen.

	Algoritmien ja ohjelmoinnin peruskurssi (AOP)	Algoritmien ja ohjelmoinnin peruskurssi (AvoinAOP)	Introduction to Programming (ItP)
Harjoitustentti	143	33	60
Ei harjoitustenttiä	40	6	8
Osallistumis-% harjoitustenttiin	78,1 %	84,6 %	88,2 %

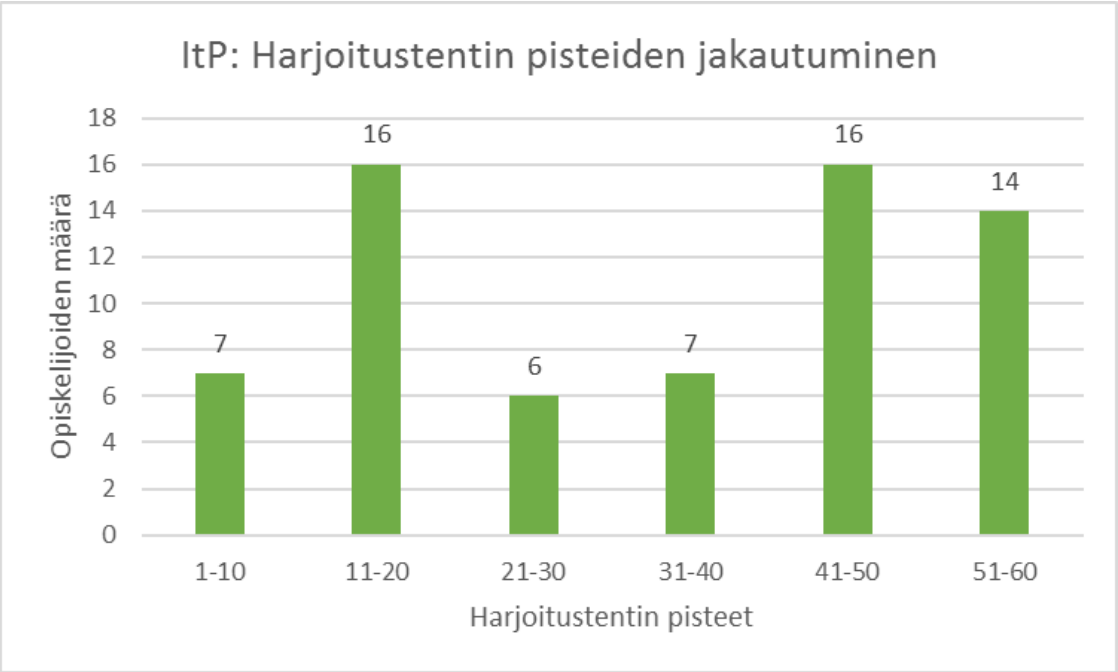
Kaikkien kurssien osalta harjoitustenteissä pärjättiin pääsääntöisesti hyvin, ja suurin osa osallistujista sai pisteitä 31-50 tai ItP:n tapauksessa 41-60. Eri kursseilla harjoitustentin pistemäärien jakautuminen näkyy kuvissa 10, 11 ja 12. Harjoitustentin jokaisesta tehtävästä sai 10 pistettä, eli kurssien AOP ja AvoinAOP harjoitustentissä oli viisi tehtävää ja kurssin ItP harjoitustentissä oli kuusi tehtävää.



Kuva 10 AOP: Harjoitustentin pisteiden jakautuminen.



Kuva 11 AvoinAOP: Harjoitustentin pisteiden jakautuminen.



Kuva 12 ItP: Harjoitustentin pisteiden jakautuminen.

Jokaisen kurssin harjoitustentin pistejakauma on kuitenkin hieman erilainen, kuten kuvista näkyy. ItP-kurssi poikkeaa AOP- ja AvoinAOP-kursseista siinä, että pisteillä 11-20 harjoitustentistä suoriutuneita on yhtä paljon kuin 41-50 pistettä saaneita. AvoinAOP-kurssilla taas kaikki saivat vähintään kahdesta tehtävästä pisteitä.

Taulukossa 6 seuraavalla sivulla on kurssien keskiarvoja. Keskiarvot on jaettu sen mukaan, osallistuivatko opiskelijat harjoitustenttiin (HT) vai ei (EiHT). Lisäksi taulukosta selviää harjoitustentin, lopputentin ja kurssin kokonaispistemäärän maksimiarvot. Nämä vaihtelevat kursseittain, joten eri kurssien välisiä keskiarvoja ei voi suoraan verrata kuin joissain tapauksissa.

Keskiarvot sekä lopputentin että kurssin kokonaispistemäärän osalta ovat harjoitustentin suorittaneilla yhtä poikkeusta lukuun ottamatta parempia kuin harjoitustenttiin ei-osallistuneilla. AvoinAOP-kurssin osalta harjoitustenttiin ei-osallistuneet ovat pärjänneet paremmin, kun katsotaan keskiarvoa. Sen sijaan, kun katsotaan mediaania, huomataan (taulukko 7 sivulla 48), että AvoinAOP-kurssilla harjoitustentin suorittaneiden mediaani on 73 pistettä ja harjoitustentin ei-suorittaneiden mediaani on 72,5.

Mediaanit tenttituloksissa ja kurssin kokonaispistemäärässä ovat kaikkien kolmen kurssin osalta korkeampia harjoitustentin suorittaneilla.

Taulukko 6 Kurssien keskiarvot harjoitustentistä, lopputentistä ja kurssin kokonaispistemäärästä.

Keskiarvo	N	Harjoitustentin tulos (max 50 pistettä)	Tentin tulos (max 90 pistettä)	Kurssin kokonais-pistemäärä (max 1560 pistettä)
AOP-HT	143	38,1	72,4	1320,3
AOP-EiHT	40	-	71,1	1185,1

Keskiarvo	N	Harjoitustentin tulos (max 50 pistettä)	Tentin tulos (max 90 pistettä)	Kurssin kokonais-pistemäärä (max 1050 pistettä)
AvoinAOP-HT	33	40,1	65,8	948,2
AvoinAOP-EiHT	6	-	72,2	896,0

Keskiarvo	N	Harjoitustentin tulos (max 60 pistettä)	Tentin tulos (max 100 pistettä)	Kurssin kokonais-pistemäärä (max 944 pistettä)
ItP-HT	60	36,6	72,0	805,9
ItP-EiHT	8	-	70,0	687,9

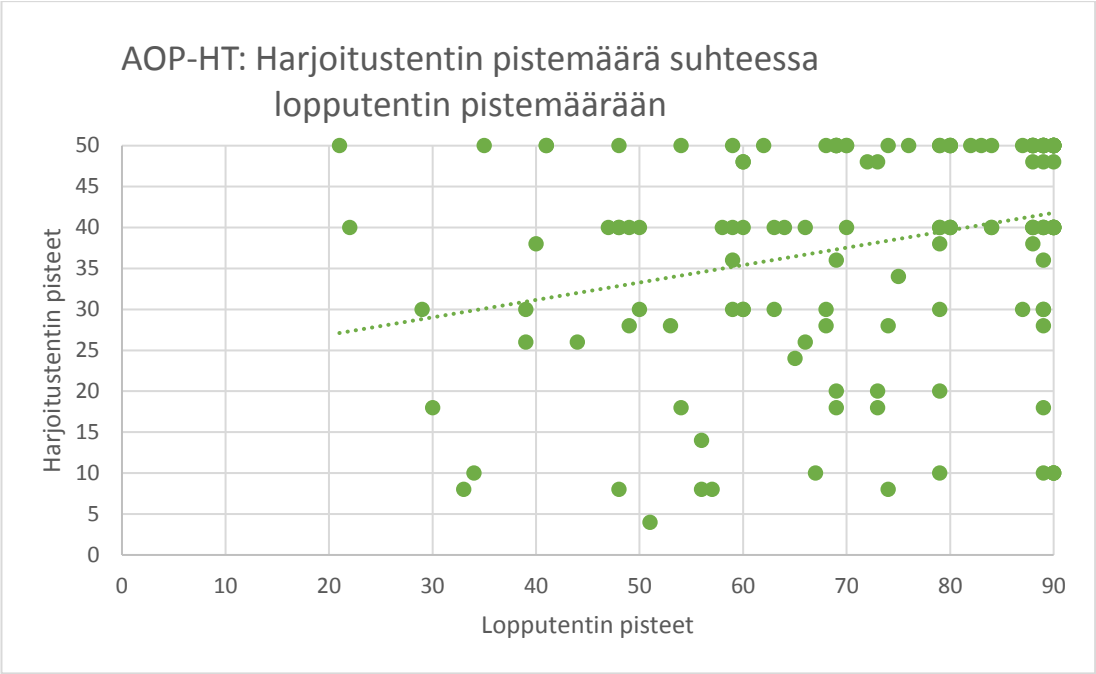
Taulukko 7 Kurssien mediaanit harjoitustentistä, lopputentistä ja kurssin kokonaispistemäärästä.

Mediaani	N	Harjoitustentin tulos (max 50 pistettä)	Tentin tulos (max 90 pistettä)	Kurssin kokonais-pistemäärä (max 1560 pistettä)
AOP-HT	143	40,0	79,0	1355,0
AOP-EiHT	40	-	77,5	1230,5

Mediaani	N	Harjoitustentin tulos (max 50 pistettä)	Tentin tulos (max 90 pistettä)	Kurssin kokonais-pistemäärä (max 1050 pistettä)
AvoinAOP-HT	33	40,0	73,0	1008,0
AvoinAOP-EiHT	6	-	72,5	946,0

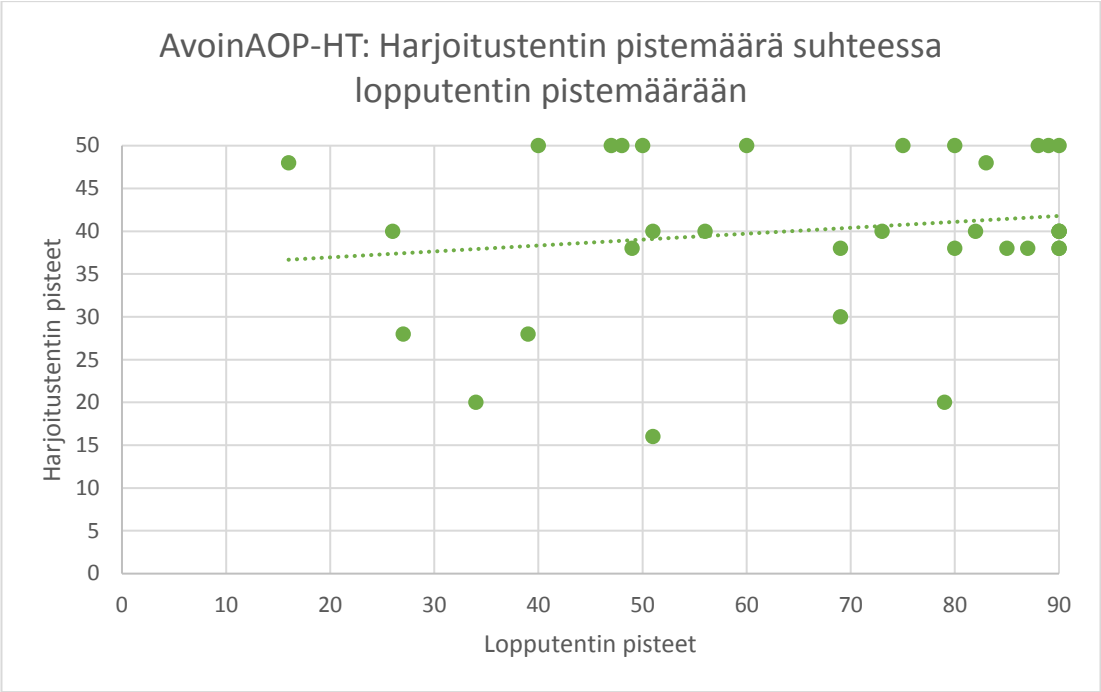
Mediaani	N	Harjoitustentin tulos (max 60 pistettä)	Tentin tulos (max 100 pistettä)	Kurssin kokonais-pistemäärä (max 944 pistettä)
ItP-HT	60	42,0	75,5	873,0
ItP-EiHT	8	-	71,0	640,0

Seuraavat kolme kuvaa (kuva 13, 14 ja 15) kuvaavat harjoitustentin pistemäärää suhteessa lopputentin pistemäärään. Kaikkien kurssien osalta löytyy hajontaa suhteessa korkeimpaan harjoitustenttipistemäärään ja lopputentin pistemäärään. Kuitenkin data jakautuu lineaarisesti nousevasti eli harjoitustentin paremmalla pistemäärällä on positiivinen vaikutus tenttitulokseen. Korrelaatio on kuitenkin heikko kursseilla AOP (0,3) ja AvoinAOP (0,2). Sen sijaan kurssilla ItP korrelaatio on vahva (0,6).

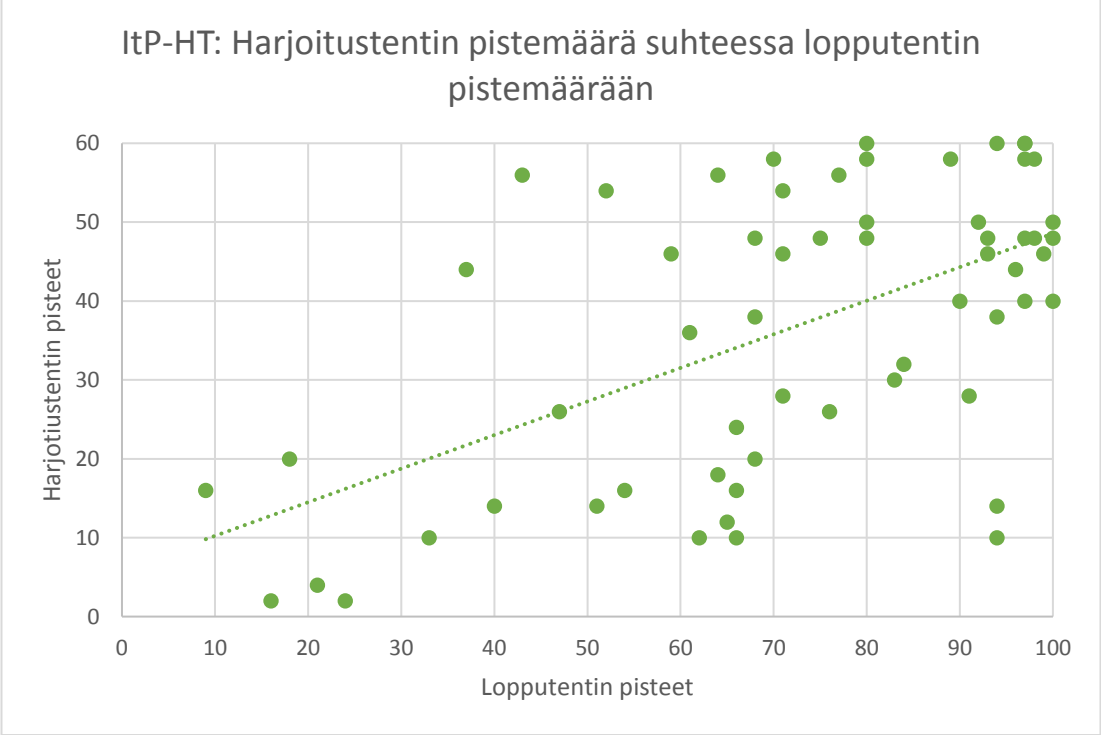


Kuva 13 AOP-HT: Harjoitustentin pistemäärä suhteessa lopputentin pistemäärään. Korrelaatio 0,3.

Kuvassa 13 kannattaa kiinnittää huomiota kuvan oikeaan laitaan eli lopputentistä täydet tai lähes täydet pisteet saaneisiin. Harjoitustentin pistemäärissä on vaihtelua 10 pisteestä 50 pisteeseen. Harjoitustentti oli vapaaehtoinen, joten on mahdollista, että osa opiskelijoista on kokeillut sitä vain yhden tehtävän verran.

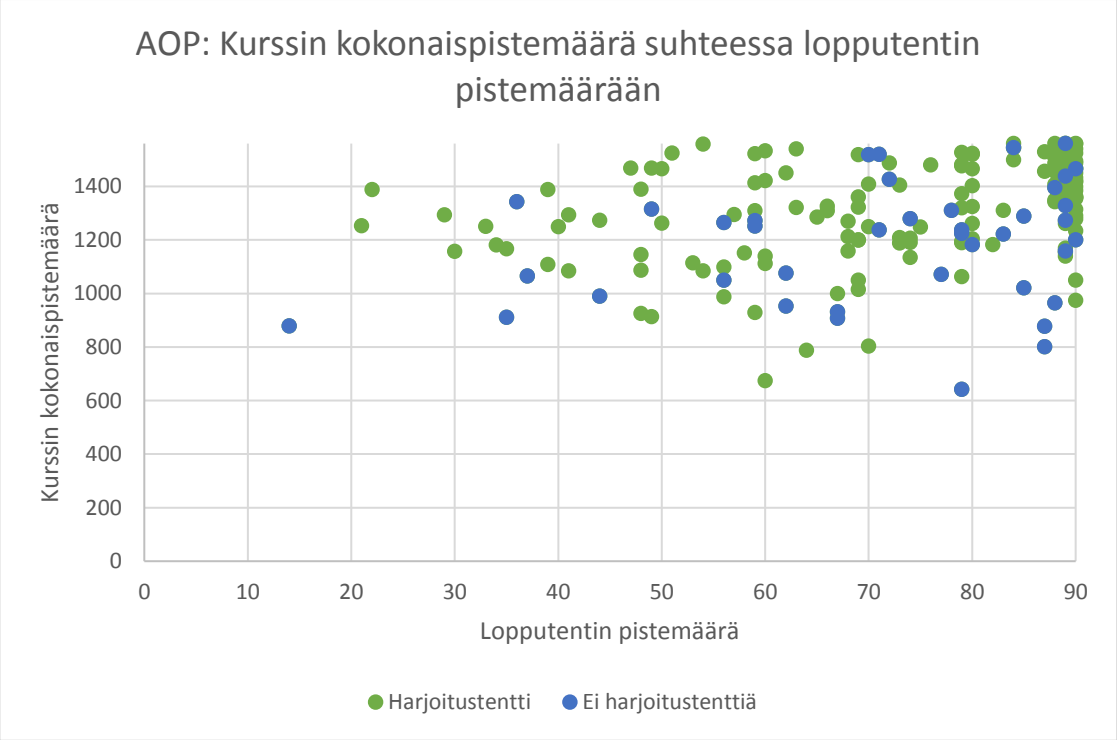


Kuva 14 AvoinAOP-HT: Harjoitustentin pistemäärä suhteessa lopputentin pistemäärään. Korrelaatio 0,2.

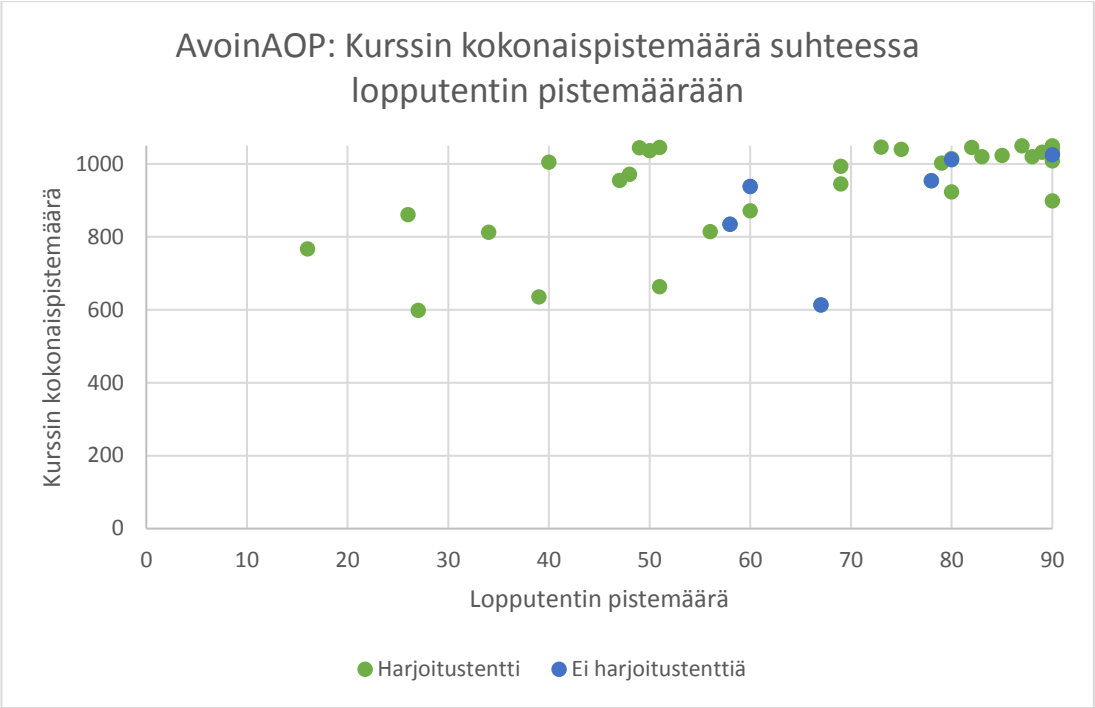


Kuva 15 ItP-HT: Harjoitustentin pistemäärä suhteessa lopputentin pistemäärään. Korrelaatio 0,6.

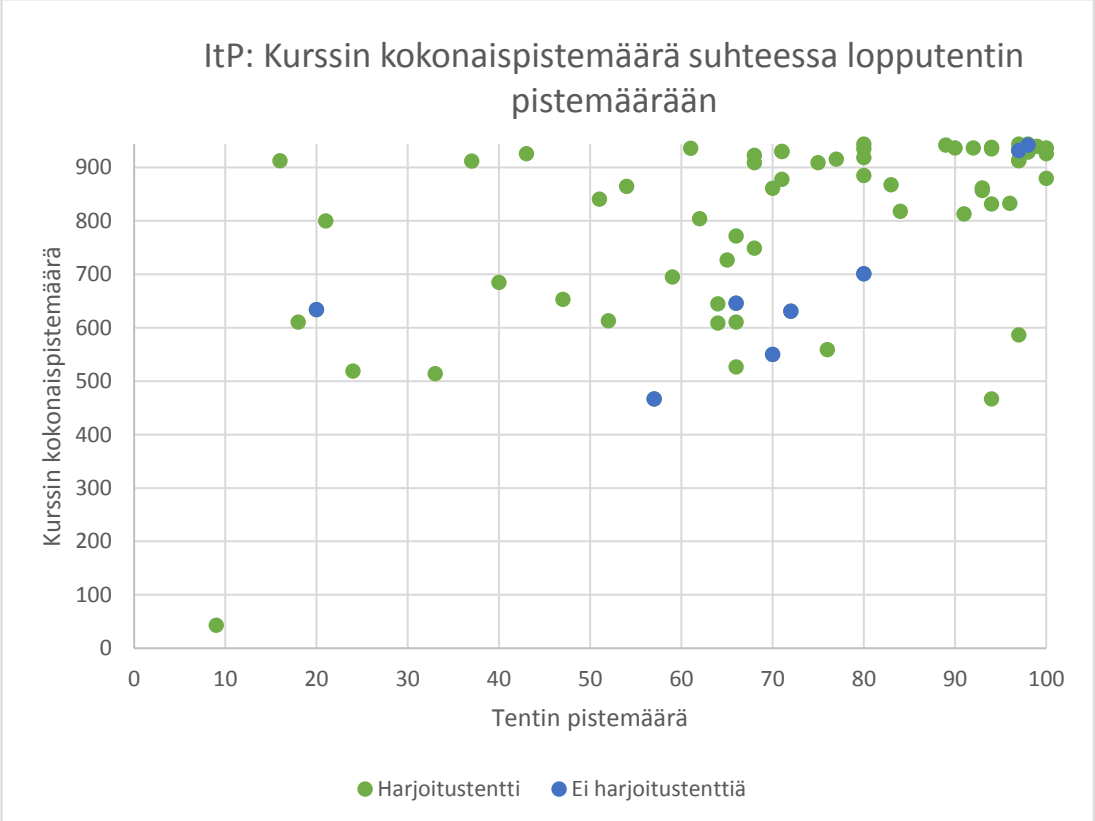
Kuvissa 16, 17, ja 18 on kuvattu kurssin kokonaispistemäärää suhteessa lopputentin pistemäärään. Näissä kaavioissa on myös mukana harjoitustenttiin osallistuneiden tulokset vihreällä ja ei-osallistuneiden tulokset sinisellä. Hyvä tenttitulos on pääsääntöisesti suhteessa myös hyvään kokonaispistemäärään, erityisesti kursseilla AvoinAOP ja ItP (korrelaatiot taulukossa 8).



Kuva 16 AOP: Kurssin kokonaispistemäärä suhteessa lopputentin pistemäärään.



Kuva 17 AvoinAOP: Kurssin kokonaispistemäärä suhteessa lopputentin pistemäärään.



Kuva 18 ItP: Kurssin kokonaispistemäärä suhteessa lopputentin pistemäärään.

ItP-kurssin osalta (kuva 18) ryhmän ItP-EiHT tulokset ovat suurelta osin kurssin kokonaispistemäärässä selkeästi alempia. Datapisteitä on kuitenkin vain kahdeksan. Tentin ja kurssin kokonaispistemäärän korrelaatio on kuitenkin tällä ryhmällä tutkittavan datan vahvin (taulukko 8). Samoin AvoinAOP-EiHT-ryhmässä on vain kuusi opiskelijaa, joten tulokseen pitää suhtautua varauksella.

Taulukko 8 Tentin ja kurssin kokonaispistemäärän korrelaatio.

	N	Korrelaatio Tentti ja kurssin kokonaispistemäärä
AOP-HT	143	0,37
AOP-EiHT	40	0,26
AOP Kaikki	183	0,34
AvoinAOP-HT	33	0,65
AvoinAOP-EiHT	6	0,55
AvoinAOP Kaikki	39	0,60
ItP-HT	60	0,51
ItP-EiHT	8	0,66
ItP Kaikki	68	0,52

AvoinAOP- ja ItP-kurssien osalta tenttipistemäärän ja kurssin kokonaispistemäärän korrelaatio on selkeä, paikoitellen jopa vahva. AOP-kurssin osalta vaihtelua on enemmän, harjoitustentin suorittaneille korrelaatio on olemassa, mutta ei niin selkeä kuin muilla kursseilla.

7 Johtopäätöksiä

Tässä työssä tutkittiin kahta tutkimuskysymystä: 1) ”Vaikuttaako ViLLE-oppimisympäristössä tehty vapaaehtoinen harjoitustentti ohjelmoinnin peruskurssin lopputentin pistemäärään?” ja 2) ”Miten harjoitustentin vaikutus eroaa tutkittavien ryhmien välillä (IT-pääaineopiskelijat, sivuaineopiskelijat, avoimen yliopiston opiskelijat)?”. Tulokset kysymykseen 1 ovat ristiriitaisia eri kurssien välillä, joten selkeää positiivista tulosta ei vapaaehtoisen harjoitustentin vaikutuksesta lopputentin pistemäärään ole nähtävissä. Huomion arvoista on kuitenkin, että vapaaehtoisen harjoitustentin suoritti kaikilla kolmella kurssilla yli 3/4 opiskelijoista, joillain jopa 88 % opiskelijoista (taulukko 5).

Kurssin lopputentin pistemäärän ja kokonaispistemäärän mediaani on harjoitustentin suorittaneilla korkeampi kuin niillä, jotka eivät harjoitustenttiä ole suorittaneet (taulukko 7). Harjoitustentti vaikuttaa kuitenkin kolmella tutkittavalla kurssilla eri tavoin, kuten luvun 6 tuloksista on huomattavissa. Harjoitustentin pistemäärä korreloi heikosti lopputentin pistemäärään kursseilla AOP (korrelaatio 0,3) ja AvoinAOP (korrelaatio 0,2). Sen sijaan kurssilla ItP, harjoitustentin pistemäärä korreloi vahvasti lopputentin pistemäärään (korrelaatio 0,6). AOP ja AvoinAOP ovat käytännössä sama kurssi, mutta niillä on vain eri kohderyhmät. ItP on hieman erilainen, kuten taulukossa 2 eriteltiin. ItP:ssä käytettävä kieli on Python ja opetuskielenä englanti. Mahdollisia syitä eroille voi jopa olla harjoitustentissä ja siinä, miten hyvin se vastaa lopputenttiä. Ilmiö voisi siis johtua siitä, vastaako ItP-kurssin harjoitustentti paremmin kurssin lopputenttiä. Pohdittavaksi jääkin, voisiko AOP- ja AvoinAOP-harjoitustenttejä muokata enemmän ItP-kurssin harjoitustenttiä vastaavaksi, jotta erot korrelaatioissa tasoittuisivat.

Harjoitustenttiin osallistumisprosentti on merkittävä jokaisella kurssilla (AOP 78,1 %, AvoinAOP 84,6 % ja ItP 88,2 %). Harjoitustentistä ei saanut pisteitä kurssin kokonaispistemäärään, ja kaikki kolme ohjelmoinnin peruskurssia ovat raskaita työmäärältään. Kursseilla on luentoja, tutoriaaleja, demonstraatioita, ViLLE-tehtäviä ja näiden lisäksi vielä itsenäistä työskentelyä. Näistä kaikista sai

pisteitä ja sen lisäksi vielä kurssin kokonaispistemäärään vaikutti luonnollisesti lopputentti. Vapaaehtoinen harjoitustentti näyttäisi kuitenkin olevan opiskelijoiden mielestä hyödyllinen, kun katsotaan osallistumisprosenttia. Harjoitustentin teko saattaa vähentää myös tenttiahdistusta tai tenttijännitystä, koska koe on samankaltainen kuin lopputentti.

AOP-kurssilla harjoitustenttiin osallistui prosentuaalisesti vähiten opiskelijoita, joka saattaa selittyä sillä, että kurssin pääasiallinen kohderyhmä on laitoksen omat pääaineopiskelijat, jotka käyttävät ViLLEä myös muilla kursseillaan. Toisin sanoen alusta itsessään voi jo olla niin tuttu, että osa opiskelijoista ei ole kokenut harjoitustenttiä tarpeelliseksi. Opiskelija on myös halutessaan voinut testata harjoitustenttiä tekemällä yhden tehtävän ja jättämällä harjoitustentin suorituksen siihen. Tämä näkyy mielestäni erityisesti kurssilla AOP.

Hyvä harjoitustenttipistemäärä ei ole tae lopputentissä menestymiselle, tämä näkyy erityisesti kursseilla AOP ja AvoinAOP (kuvat 13 ja 14). Syitä tähän voi olla esimerkiksi harjoitustentin teko ryhmässä tai tenttijännitys. Toisaalta heikot pisteet harjoitustentissä eivät näillä kahdella kurssilla automaattisesti tarkoittaneet huonoa tulosta lopputentissä, joka oli jo nähtävissä siinä, että korrelaatio oli heikko harjoitustentin tuloksen ja lopputentin tuloksen välillä.

Kurssilla AOP harjoitustentin tehneillä lopputentin pistemäärä korreloi kurssin kokonaispistemäärän kanssa (0,37). Tulos ei ole vahva, mutta korrelaatio on olemassa. Sen sijaan kurssilla AvoinAOP korrelaatio lopputentin ja kurssin kokonaispistemäärän välillä on harjoitustentin tehneillä 0,65 ja muilla 0,55. Toisin sanoen hyvät pistemäärät kurssin aikana ja tehty harjoitustentti korreloi myös hyvän lopputentin pistemäärän kanssa. ItP-kurssilla vahvin korrelaatio (0,66) on heillä, jotka eivät tehneet harjoitustenttiä. Tuloksesta on huomioitava, että N=8.

Muutenkin jokaisesta tuloksesta on otettava huomioon se, että harjoitustentin osallistumisprosentin ollessa jokaisella kurssilla korkea, se vaikuttaa tutkittavaan dataan ja siihen, että harjoitustenttiin osallistumattomien osuus on aina pieni suhteessa osallistuneisiin. Pienemmillä kursseilla, kuten AvoinAOP ja

ItP tämä näkyy erityisesti, koska harjoitustenttiin jätti osallistumatta AvoinAOP-kurssilla 6 henkilöä ja ItP-kurssilla 8 henkilöä. Mielenkiintoista olisikin tulosten vertailu kurssiin, jolla ei ole käytössä minkäänlaista harjoitustenttiä.

Tuloksista on myös nähtävissä sama kuin Peter Reuterin ja Stacey Parkerin (2015) tuloksissa, jotka esiteltiin luvussa 3.3. Harjoituskokeen suorittaminen tai sen suorittamatta jättäminen ei vaikuta korkeimman arvosanan (tuloksissa korkeimpien pistemäärien) saaneiden määrään.

Palaan vielä kurssin työmäärään. Kuvassa 9 esittelin aineiston rajauksen yhteydessä kurssien ViLLE-alueelle kirjautuneiden määrän ja tenttiin osallistuneiden määrän. Tenttiin osallistumisprosentit olivat eri kursseilla seuraavat: AOP 71,2 %, AvoinAOP 52,7 % ja ItP 70,8 %. AvoinAOP-kurssille osallistuvat tulevat avoimen yliopiston kautta ja suorittavat kurssia esimerkiksi työn ohella. Kurssin korkea työmäärä voi vaikuttaa siihen, että lähes puolet AvoinAOP-kurssille alussa osallistuneista opiskelijoista ei koskaan osallistunut tenttiin.

8 Yhteenveto

Ohjelmointia opetellaan yhä enemmän kaikilla kouluasteilla. Tämä tuo mukanaan uusia oppimisympäristöjä, ja niiden käyttöä on harjoiteltava. Sama pätee myös kolmeen tutkittavana olleeseen ohjelmoinnin johdantokurssiin, joiden lopputentteihin osallistui 290 opiskelijaa. Yli ¾ kursseille osallistuneista, jotka osallistuivat myös lopputenttiin, tekivät vapaaehtoisen harjoitustentin ViLLE-oppimisympäristössä.

Tutkielmani tutkimuskysymykset olivat:

1. Vaikuttaako ViLLE-oppimisympäristössä tehty vapaaehtoinen harjoitustentti ohjelmoinnin peruskurssin lopputentin pistemäärään?
2. Miten harjoitustentin vaikutus eroaa tutkittavien ryhmien välillä (IT-pääaineopiskelijat, sivuaineopiskelijat, avoimen yliopiston opiskelijat)?

Vapaaehtoinen harjoitustentti ja sen tekeminen vaikutti ristiriitaisesti kurssin lopputentin pistemäärään. Kaikkien kurssien osalta harjoitustentin suorittaneiden opiskelijoiden mediaanitulokset lopputentissä olivat korkeammat kuin harjoitustentin väliin jättäneillä opiskelijoilla. Harjoitustentin vaikutus ei kuitenkaan ollut merkittävä kursseilla Algoritmien ja ohjelmoinnin peruskurssi (pääasiassa IT-pääaineopiskelijat) ja Avoimen yliopiston Algoritmien ohjelmoinnin peruskurssi. Sen sijaan sivuaine- ja vaihto-opiskelijoille tarkoitetulla Introduction to Programming -kurssilla harjoitustentin pistemäärän korrelaatio lopputentin pistemäärä oli vahva.

Tässä työssä tutkin samalta syksyltä kolmea ohjelmoinnin perusteiden kurssia. Kiinnostavaa olisi tutkia useamman vuoden ajalta näitä kursseja tai edes yhtä niistä. Ovatko tulokset muuttuneet kurssin alkuajoista tässä toteutusmuodossa tähän päivään mennessä, ja jos ovat, millä tavalla? Lisäksi kursseja ja harjoitustenttejä voisi tutkia vielä tarkemmin. Harjoitustenteistä voisi esimerkiksi tutkia, kuinka kauan opiskelijat käyttivät harjoitustenttiin ja sen eri tehtäviin aikaa, ja miten harjoitustenttiin valitut tehtävät vastasivat kurssin lopputentissä olleita tehtäviä.

Lopuksi palaan alussa esiteltyyn Kayn, Doddin ja Simen (1970) kirjaan ohjelmoidusta opetuksesta 1960-luvun lopulta. Vaikka aikaa on kulunut 50 vuotta, perustavoitteemme on edelleen sama – opettaa opiskelijoita parhaamme mukaan käytössä olevia apuvälineitä hyödyntäen ja kehittäen.

”Olemme puhuneet siitä, kuinka tietokoneet sekä antavat opetettavaa materiaalia oppilaille, että pitävät kirjaa heidän oppimissuorituksistaan. Itse asiassa tietokone suorittaakin koko joukon tehtäviä, jotka meidän pitäisi tuntea, sillä ne auttavat meitä ohjaamaan opiskelijan oppikurssia niin, että hän itse saa suurimman mahdollisimman hyödyn ja että itse opetusmenetelmä kehittyy kokemuksen myötä. Tuleviakin polvia on opetettava!” (Kay ym. 1970, 149-150.)

LÄHTEET

Apiola, M-V. 2013. *Creativity-Supporting Learning Environments: Two Case Studies on Teaching Programming*. https://helda.helsinki.fi/bitstream/handle/10138/40146/apiola_dissertation.pdf?sequence=1.

Bloom, B. S., Anderson, L. W. & Krathwohl, D. R. 2001. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's Taxonomy of educational objectives*. Abridged ed. New York: Longman.

Crisp, G. 2011. *Teacher's Handbook on e-Assessment*. Transforming Assessment – An ALTC Fellowship Activity, vol. 18. http://www.bezaspeaks.com/eassessmentafrica/Handbook_for_teachers.pdf [viitattu 1.5.2019.]

Esteves, M., Fonseca, B., Morgado, L. & Martins, P. 2009. *Using Second Life for Problem Based Learning in Computer Science Programming*. Journal of Virtual Worlds Research, vol. 2, no. 1. Sivut 2-25. https://journals.tdl.org/jvwr/index.php/jvwr/article/view/419.

Gaspar, A., Langevin, S. & Boyer, N. 2009. *Facilitating students-driven learning of computer programming with technology*. Kirjassa *Information technology and constructivism in higher education: progressive learning frameworks*. Payne, CR. (toim.). Hershey, PA: Information Science Reference. Sivut 262-275.

Gehrig, E. 2017 *What is a Practice Exam and What Can it Do for Me?* https://blog.tesu.edu/what-is-a-practice-exam-and-what-can-it-do-for-me [viitattu 17.3.2019].

Griffin, P. & Care E. 2014. *Introduction: Assessment is for teaching*. Kirjassa *Assessment for teaching*. Griffin, P. (toim.). Cambridge: Cambridge University Press. Sivut 1-12.

Hiltunen, L. 2012. *Verkko-opetuksen suunnittelun tehostaminen*. Tietojenkäsittelytiede 34. http://www.cse.tkk.fi/fi/tkt-lehti/a34/hiltunen.pdf.

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2009. *Tutki ja kirjoita*. Helsinki: Tammi.

Jonsson, H. 2015. *Using flipped classroom, peer discussion, and just-in-time teaching to increase learning in a programming course*. 2015 IEEE Frontiers in Education Conference (FIE). https://doi.org/10.1109/FIE.2015.7344221.

Kaila, E. 2018. *Utilizing educational technology in computer science and programming courses: Theory and practice*. Turku Centre for Computer Science. http://www.utupub.fi/handle/10024/144535.

Kaila, E. & Kurvinen, E. 2018. *ViLLE – Opettajan kirja*. https://www.oppimisanalytiikka.fi/sites/default/files/documentation/TheBookOfViLLE_fi.pdf [viitattu 12.1.2019].

Kaila, E., Laakso, M.-J., Rajala, T. & Kurvinen, E. 2018. *A model for gamifying programming education: University-level programming course quantified*. 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). http://dx.doi.org/10.23919/MIPRO.2018.8400129.

Karjalainen, A. 2001. *Tentin teoria*. Oulun yliopistopaino. http://tievie.oulu.fi/arvioinnin_abc/liitetiedostot/tentin_teoria_vaitoskirja.pdf .

Kay, H., Dodd, B. & Sime, M. 1970. *Opetuskoneet ja ohjelmoitu opetus*. Hki: Weilin + Göös.

Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, JH. & Crawford, K. 2000. *Problem-Based Learning for Foundation Computer Science Courses*. Computer Science Education, vol 10, no. 2. Sivut 109-128. http://dx.doi.org/10.1076/0899-3408(200008)10:2;1-C;FT109.

Laakso, M.-J., Kaila, E. & Rajala, T. 2018. *ViLLE – collaborative education tool: Designing and utilizing an exercise-based learning environment*. Education and Information Technologies, vol 23, no. 4, pages 1656-1676. http://dx.doi.org/10.1007/s10639-017-9659-1.

Lehtinen, E., Vauras, M. & Lerkkanen M-K. 2016. *Kasvatuspsykologia*. Jyväskylä: PS-kustannus.

Maher, M.L., Latulipe, C., Lipford, H. & Rorrer, A. 2015. *Flipped Classroom Strategies for CS Education*. SIGCSE '15 Proceedings of the 46th ACM Technical Symposium on Computer Science Education, sivut 218-223, ACM New York. https://dl.acm.org/citation.cfm?id=2677252.

Neisser, U. 1976. *Cognition and Reality: Principles and implications of cognitive psychology*. San Francisco: Freeman.

Oliver, R. & Williams, R.L. 2005. *Direct and Indirect Effects of Completion Versus Accuracy Contingencies on Practice-Exam and Actual-Exam Performance*. Journal of Behavioral Education, June 2005, Volume 14, Issue 2, sivut 141–152. https://doi.org/10.1007/s10864-005-2707-8.

Oppimisanalytiikan keskus. 2018. *ViLLE*. https://oppimisanalytiikka.fi/fi/ville [viitattu 17.5.2018].

OPS2014. Perusopetuksen opetussuunnitelman perusteet. 2014. Helsinki: Opetushallitus.

Pavlovic, M., Awwal, N., Mountain, R. & Hutchinson, D. 2014. *Conducting assessments*. Kirjassa *Assessment for teaching*. Griffin, P. (toim.). Cambridge: Cambridge University Press. Sivut 58-68.

Rajala, T., Kaila, E., Lindén, R., Kurvinen, E., Lokkila, E., Laakso M.-J. & Salakoski, T. 2016. *Automatically assessed electronic exams in programming courses*. Teoksessa *Proceedings of the Australasian Computer Science Week*. http://doi.org/10.1145/2843043.2843062.

Reuter, P. & Parker, S. 2015. *Low-Stakes Online Practice Exams Increase Students' Performance*. Teoksessa *Formal Exams And Overall Success In Science Courses. SoTL Commons Conference*. 123. https://digitalcommons.georgiasouthern.edu/sotlcommons/SoTL/2015/123.

Rinne, R., Kivirauma J. & Lehtinen, E. 2010. Johdatus kasvatustieteisiin. Helsinki: WSOYpro Oy.

Savery, JR. 2015. *Overview of Problem Based Learning: Definition and Distinctions*. Kirjassa *Essential Readings in Problem-Based Learning*. Ertmer PA, Hmelo-Silver CE, Leary H & Walker A, (toim.). West Lafayette: Purdue University Press. Sivut 5-15.

Schommer, M. 1990. *Effects of beliefs about the nature of knowledge on comprehension*. Journal of Educational Psychology, 82 (3). Sivut 498-504.

Scott, T. 2003. *Bloom's Taxonomy Applied to Testing in Computer Science Classes*. Journal of Computing Sciences in Colleges archive, Volume 19 Issue 1, October 2003. Sivut 267-274.

Siltari, E. 2018. *Verkko-opetus ja sen suunnittelu – käytännön näkökulma*. Luento 6.3.2018.

Suominen, R. & Nurmela, S. 2011. *Verkko-opettaja*. Helsinki: WSOYpro.

Toivola, M. 2015. *Flipped learning*. https://www.utu.fi/fi/sivustot/koulutus-ja-kehittamispalvelut/oikeasti-oppimaan/paikalliset-toimijat/tieto-ja-viestintateknologian-hyodyntaminen/flipped-learning/Sivut/home.aspx [viitattu 17.3.2019].

Toivola, M., Peura P. & Humaloja M. 2017. *Flipped learning: Käänteinen oppiminen*. Helsinki: Edita.

Toivola, M. 2019. *Käänteinen oppiminen – kääntyykö koulutyö päälaelleen?* teoksessa Tossavainen T. & Löytönen M. (toim), Sähköistyvä koulu: Oppiminen ja oppimateriaalit muuttuvassa tietoympäristössä. Suomen tietokirjailijat ry, Helsinki, sivut 98-116.

Turun yliopisto. 2017a. *Avoimen yliopiston Nettiopsu: Algoritmien ja ohjelmoinnin peruskurssi 6 op*. https://nettiopsu.utu.fi/avoin/ilmoittautuminen/hakukohteet/26668 [viitattu 12.1.2019].

Turun yliopisto. 2017b. *Opinto-oppaat: BIOI2250 Introduction to Programming 5–6 op*. https://nettiopsu.utu.fi/opas/opintojakso.htm?rid=31509&idx=2&uiLang=fi&lang=en&lvv=2017 [viitattu 12.1.2019].

Turun yliopisto. 2017c. *Opinto-oppaat: TKO_2038 Algoritmien ja ohjelmoinnin peruskurssi 5–6 op*. https://nettiopsu.utu.fi/opas/opintojakso.htm?rid=26008&idx=1&uiLang=fi&lang=fi&lvv=2017 [viitattu 12.1.2019].

Turun yliopisto. 2019. *Sähköinen tenttipalvelu: opettajan ohje*. https://utuguides.fi/sahkoinententti [viitattu 22.2.2019].

Tynjälä, P. 1999. *Oppiminen tiedon rakentamisena: Konstruktivistisen oppimiskäsityksen perusteita*. Helsinki: Kirjayhtymä.

Vanderbilt University Center for Teaching. *Bloom's Taxonomy*. https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy/ [viitattu 22.2.2019].

Vihavainen, A., Airaksinen, J. & Watson, C. 2014. *A Systematic Review of Approaches for Teaching Introductory Programming and Their Influence on Success*. ICER '14 Proceedings of the tenth annual conference on International computing education research. Sivut 19-26. http://dl.acm.org/citation.cfm?id=2632349.

Viitala, T. & Lehtelä, P-L. 2009. *Verkon varassa*. Kirjassa *Verkon varassa. Opetuksen pedagoginen kehittäminen verkkoympäristössä*. Ihanainen, P., P. Kalli & K. Kiviniemi (toim.). Jyväskylä: Jyväskylän Ammattikorkeakoulu. Sivut 51-62.

Watson, C. & Li, F. W. 2014. *Failure rates in introductory programming revisited*. Proc. Innovation and Technology in Computer Science Education (ITiCSE). ACM. http://dx.doi.org/10.1145/2591708.2591749.