Vertaisoppiminen ja parityöskentely olio-ohjelmoinnin opintojaksolla sekä luento-opetuksen kehittäminen

Erno Vanhala

Tuotantotalouden tiedekunta
Ohjelmistotuotanto ja tiedonhallinta

Tiivistelmä

Tässä työssä käsitellään Lappeenrannan teknillisen yliopiston tietotekniikan koulutusohjelman Olio-ohjelmoinnin opintojakson kehittämistä. Lähtökohtana on opintojakson nykytila, jossa opetusta tapahtuu niin lähiopetuksena kuin myös verkossa tapahtuvana oppimisena virtuaalisissa oppimisympäristöissä. Opintojakso on saanut hyvää palautetta vuosi vuodelta, mutta opiskelijoiden heikko osallistuminen luennoille ja harjoituksiin on herättänyt kysymyksen luentojen tarpeellisuudesta ja harjoitusten roolin muuttamisesta.

Opintojakson luennot on nauhoitettu ja tarjottu verkossa opiskelijoiden katsottaviksi, mutta tähän mennessä on puhuttu 40-90 minuutin luentokokonaisuuksista, jotka ovat voineet sisältää useita eri asioita. Tämä on saanut kritiikkiä osakseen ja toiveissa on ollut pieniä asiakokonaisuuksia painottavia ”täsmäiskuja”.

Harjoitukset ovat olleet vapaaehtoisia, eikä niistä ole saanut pisteitä, joten opiskelijoiden intensiteetti osallistua niihin ei ole ollut suuri. Tämän opiskelijat ovat myös raportoineet kurssipalautteessa.

Opintojakson kehityksessä lähiopetuksena hoidetut luennot päätettiin jättää kokonaan pois ja tuoda tilalle kymmeniä lyhyitä opetusvideoita opintojaksolla käsiteltävistä asioista. Opiskelija on velvoitettu katsomaan viikon videot ennen harjoituksia ja näiden pohjalta päästään harjoituksissa opettelemaan käytännössä videoilla käsiteltyjä asioita. Harjoitusten suorittamisesta annetaan myös pisteitä kurssisuoritukseen. Harjoituksissa ja harjoitustyöntekemisessä kannustetaan parityöskentelyyn.

Johdanto

Tässä työssä käsitellään Olio-ohjelmoinnin opintojakson kehitystä. Opintojakso on ollut tietotekniikan koulutusohjelmassa jo yli kymmenen vuotta ja sitä on luennoinut useita eri luennoitsijoita. Itse olen pitänyt opintojaksoa syksystä 2010.

Opintojakso on tarkoitettu toisen vuoden opiskelijoille ja se on heidän kolmas ohjelmointikurssinsa. Siinä missä ensimmäisen vuoden kursseilla Ohjelmoinnin perusteet ja Käytännön ohjelmointi ohjelmoidaan proseduraalisesti, opetellaan toisen opiskeluvuoden aluksi olio-ohjelmointia. Tietotekniikan koulutusohjelmamme on valinnut proseduraalisen lähestymistavan ohjelmointiin, joten oliosuuntautuminen tulee vastaan toisena opiskeluvuotena. Trendi on ollut viime vuosina siirtyä jo ensimmäisellä ohjelmointikurssilla oliopohjaiseen ohjelmointiin (Lewis 2000; Vilner, Zur, ja Gal-Ezer 2007), mutta tietotekniikan koulutusohjelmassa tätä ei ainakaan vielä ole nähty tarpeelliseksi.

Opintojaksolla opetellaan myös koodin kirjoittaminen ja suunnittelutyö englanniksi, vaikka itse opintojakso kuitenkin luennoidaan suomeksi. Opintojakso ei rajoitu pelkästään ohjelmoinnin opetteluun, vaan mukana on myös suunnittelua, erilaisten ohjelmien käyttöä ja dokumentaation kirjoittamista.

Hyvästä palautteesta huolimatta ongelmia opintojaksolla on muutamia. Opiskelijat osallistuvat heikosti niin harjoituksiin kuin luennoillekin, ryhmätyökalut eivät ole olleet vielä tarpeeksi hyvin hallinnassa ja paperinen tentti on myös saanut kritiikkiä osakseen. Tässä työssä paneudutaan näiden ongelmakohtien poistamiseen, kun kurssia nyt uusitaan kesällä 2014 rankasti ohjelmointikielen ja periodin pituuden vaihduttua.

Opintojakson historia ja nykytila

Eri luennoitsijat ovat painottaneet eri asioita opintojaksolla. Opintojakson muokkaaminen alkoi tekemällä luentokalvojen tueksi myös oikeita koodiesimerkkejä, joita pystyi ohjelmoimaan luennoilla. Myös luentokalvot uusittiin 2010 uuteen uskoon, jotta värimaailma vastaa yliopiston nykyistä tyyliä ja samalla myös asiasisältö tuli paremmin ajan tasalle. Tarkempaa kuvausta opintojakson kehityksestä löytyy taulukosta 1.

Taulukko 1: Olio-ohjelmoinnin opintojakson kehitys viimeisen neljän vuoden ajalta.

2010	Opintojaksolle luotiin uudet luentokalvot ja niitä tukevat ohjelmointiesimerkit. Luennot nauhoitettiin. Opintojakso luennointiin Linux-ympäristössä pelkällä tekstieditorilla, mutta harjoitukset olivat edelleen Windows-mikroluokassa.
2011	Opintojakson harjoitukset siirrettiin Linux-luokkaan ja ohjelmointiympäristöksi valittiin niin

	luennoille kuin harjoituksiin NetBeans. Opiskelijat saivat halutessaan käyttää myös muita ympäristöjä. Luennoille lisättiin myös demo graafisesta käyttöliittymästä Qt:llä. Luentonauhoitteet tarjottiin opiskelijapalautteen pohjalta sekä videona että mp3-äänitiedostoina. Ylimääräinen esseepohjainen bonustehtävä esiteltiin opintojaksolle.
2012	Opintojakso pysyi lähes muuttumattomana verrattuna edelliseen vuoteen. Harjoitustyön aihe oli näistä vuosista kaikkein paras ja tuki oppimista hyvin.
2013	Luentonauhoitteet siirtyivät YouTubeen, joten opiskelijoilta ei vaadittu enää erillistä videoiden lataamista, vaan he pystyivät katsomaan videot vaikka suoraan kännykällä. Videoiden ääniraidan pystyi kuitenkin edelleen lataamaan YouTubesta mp3-tiedostona, jos opiskelija näin halusi. Harjoitustyön aihe oli huono ja se sai myös negatiivista palautetta kurssipalautteessa. Opintojakson alussa tarjottiin myös luentobingo, jonka opiskelijat olisivat voineet täyttää opintojakson edetessä. Kukaan ei palauttanut kuponkia. Graafisen käyttöliittymän tekemistä Qt:lla esiteltiin laajemmin kuin aiempina vuosina ja opiskelijoille esiteltiin myös NetBeanssin integroiminen git-versionhallinnalla BitBucket-palvelun tarjoamiin varastoihin. Versionhallinnan käyttö ei ollut pakollista, mutta siitä palkittiin bonuspisteellä.

Opintojakso on pitänyt sisällään 14 viikkoa luentoja ja harjoituksia, itseopiskelua Viope-oppimisympäristössä, harjoitustyön sekä tentin. Tämän lisäksi on mahdollista saavuttaa bonuspisteitä lukemalla artikkeli ja peilaamalla sitä esseessä omaan ohjelmointiosaamiseen.

Opintojaksolla opetusta on ollut tarjolla niin lähiopetuksena kuin itsenäisenä verkko-opetuksenakin. Vaikka voidaankin argumentoida kaiken opetuksen olevan sulautettua, on Olio-ohjelmoinnin opintojaksolla kuitenkin käytetty sulautettua oppimista sen nykyisessä määritelmässään, jossa opetusta tarjotaan sekä lähiopetuksena että oman aikataulun mukaisesti toimivana opetuksena verkkoympäristössä (Moskal, Dziuban, ja Hartman 2013; Wai ja Seng 2014). Tietotekniikka on alana muutenkin sellainen, että paljon opetuksesta tapahtuu verkkotyökaluilla.

Kurssipalautteen pohjalta muodostettu kuva opintojakson toimivuudesta

Kurssipalaute on ollut tähän mennessä vähintäänkin hyvää. Taulukoissa 2 ja 3 on opiskelijapalautteesta saatuja arvosanoja opintojakson eri elementeille.

Taulukko 2: Opintojakson osien haasteellisuus (1=vaikea, 5=helppo).

	2010 (N=19)	**2011 (N=18)**	**2012 (N=19)**	**2013 (N=14)**	**Keskiarvo**
Luennot	4,12	3,61	3,63	4,00	3,84
Luentokalvot	3,82	3,77	3,68	4,14	3,85
Luentonauhoitteet	3,63	3,56	3,83	4,15	3,79

Esimerkkiohjelmat	3,59	3,56	3,32	3,86	3,58
Harjoitukset	3,29	3,00	3,16	2,23	2,92
Viope-tehtävät	3,77	3,17	3,16	3,93	3,51
Harjoitustyö	3,29	2,72	2,58	2,64	2,81
Alan kirjallisuus	3,18	3,00	2,88	3,23	3,07
Internetin C++-manuaalit yms.	3,77	3,13	3,37	3,31	3,40

Taulukko 3: Opintojakson osien hyödyllisyys (1=hyödytön, 5=hyödyllinen).

	2010 (N=19)	**2011 (N=18)**	**2012 (N=19)**	**2013 (N=15)**	**Keskiarvo**
Luennot	3,94	3,65	4,11	3,87	3,89
Luentokalvot	4,06	3,53	3,68	3,80	3,77
Luentonauhoitteet	3,24	3,82	4,17	4,36	3,90
Esimerkkiohjelmat	3,88	4,35	4,21	4,20	4,16
Harjoitukset	3,41	3,50	3,79	3,36	3,52
Viope-tehtävät	3,89	4,24	3,68	3,67	3,87
Harjoitustyö	4,06	4,29	4,47	4,67	4,37
Alan kirjallisuus	3,35	3,54	3,00	3,57	3,37
Internetin C++-manuaalit yms.	3,88	4,06	4,11	4,20	4,06

Palautteen pohjalta voidaan tehdä pari päätelmää. Harjoitustyö on selkeästi opintojakson vaikein elementti, mutta se on myös hyödyllisin. Tämä sama ilmiö on nähtävissä myös muilla ohjelmointikursseilla. Ohjelmoinnin – kuten monen muunkin asian – oppii vain tekemällä itse.

Toinen merkillepantava asia on luentojen ja luentonauhoitteiden hyödyllisyys. Luentonauhoitteet on koettu hyödyllisemmäksi vuosi vuodelta kun taas luentojen hyödyllisyys on sahannut ylös ja alas. Arvostus on kuitenkin ollut korkea siihen nähden, että välillä luennoilla on istunut vain muutama opiskelija.

Opintojaksolta kerätty laadullinen palaute on ollut pääsääntöisesti hyvin positiivista, mutta myös rakentavia kehitysehdotuksia on tullut. Mitään haukkumista tai solvaamista palautteessa ei ole ollut. Palaute on kerätty anonyymisti.

Selkeät ongelmakohdat

Opintojaksolla on selkeästi neljä elementtiä, joissa on kehittämisen varaa:

- opiskelijoiden heikko osallistuminen viikkoharjoituksiin
- ryhmätyötyökalujen käytön osaamisen puute
- opiskelijoiden vähäinen osallistuminen luennoille
- paperinen tentti.

Näistä suurin ongelmakohta käsittää opiskelijoiden huonon osallistumisen viikkoharjoituksiin. Moni myös myöntää kurssipalautteessa, että harjoitustyö oli, virtuaalisessa oppimisympäristössä (Viope) tehtyjen pienten tehtävien lisäksi, ainut ohjelmointityö, jonka he suorittivat opintojakson aikana. Harjoitusten tarkoitus on opettaa luennolla käsiteltyjä tekniikoita käytännössä ja valmistaa opiskelija harjoitustyön tekoon. Kun harjoituksissa ei käydä, eikä niitä omalla ajalla tehdä, tulee harjoitustyön tekemisestä vaikeampaa ja tämä johtaa siihen, että yli puolet palautetuista harjoitustöistä lähtee bumerangina takaisin.

Toinen ongelma on ryhmätyötyökalujen käytön osaamisen puute, joka pahimmillaan ilmenee niin, että harjoitustyötä parityönä tekevät opiskelijat lähettelevät koodeja ja dokumentteja toisilleen sähköpostilla. Opiskelijat opettelevat Subversion-versionhallinnan käytön edellisellä Käytännön ohjelmoinnin opintojaksolla, mutta Olio-ohjelmointiin tulee myös opiskelijoita, jotka eivät ole Käytännön ohjelmointia käyneet tai eivät ole vielä sisäistäneet versionhallintaa.

Kolmantena, joskin vähäisempänä, asiana nostan esiin opiskelijoiden vähäisen aktiivisuuden luennoilla. Tämä näkyy sekä vähäisinä kysymyksinä – mikä ei ole mitenkään harvinaista suomalaisessa yliopistomaailmassa – mutta myös vähäisenä osallistumisena – mikä ei sekään ole välttämättä kovin uusi ilmiö. Tämä kulminoitui loppusyksystä 2013, kun luennolla oli läsnä vain kaksi opiskelijaa, jotka eivät esittäneet yhtään kysymystä koko luennon aikana. Osasyynsä tähän luonnollisesti aiheuttaa luentojen tarjoaminen YouTubessa, jolloin opiskelijalla on mahdollisuus nähdä luento, vaikkei hän tulisikaan fyysisesti paikalle.

Neljäntenä ongelmakohtana opiskelijat ovat kritisoineet kurssipalautteessa tenttiä, koska siinä joutuu ohjelmoimaan paperille, eikä se ole mielekästä. Tenttiä on pidetty ainoana mahdollisuutena tarkistaa, että opiskelija todella osaa ohjelmoida itse. Harva kuitenkaan muistaa syntaksia täydellisesti, joten tentissä paperille ohjelmointi on koettu lähinnä tyhmäksi.

Kehitysmenetelmät

Koska opintojakso on klassinen tietotekniikan koulutusohjelman luento-harjoitukset-harjoitustyö-tentti-opintojakso, ei vertaisoppimiselle ole annettu juurikaan painoarvoa, vaikka sitä voitaisiin selkeästi käyttää hyväksi opintojaksolla. Samalla, kun mukaan tuodaan vertaisoppimista, tarvitaan myös niitä tukemaan ryhmätyöskentelyä mahdollistavia työkaluja. Tästä syystä tavoitellaan opintojaksolla kahta asiaa: 1) Kannustamaan opiskelijoita tekemään töitä yhdessä, esimerkiksi, parikoodaamalla. Tätä varten esitellään työvälineitä, jotka mahdollistavat ryhmätyöskentelyn. 2) Parannuksia virtuaalisiin opetusmetodeihin ja ajankäytön siirtämistä luennoista harjoituksiin – konkreettiseen oppimiseen.

Vertaisoppiminen

Käytän tässä työssä kahta teoreettista viitekehystä. Ensimmäinen niistä on vertaisoppiminen, jonka avulla opiskelijat opettavat toisiaan, eikä kaikki oppimisvastuu tapahdu pelkästään opiskelija-opettaja akselilla. Vertaisoppiminen tarkoittaa opetusmenetelmää, jossa opiskelijat eivät ole vain passiivisia opetuksen vastaanottajia vaan osallistuvat myös itse opettamiseen käymällä lävitse ongelma- tai case-esimerkkejä (Boud ja Lee 2005; Carbery ja Hegarty 2011). Tämä on todettu erityisen toimivaksi ympäristössä, jossa edellä mainitut ongelma- tai case-esimerkit ovat arkipäivää, kuten kaupallisen alan tai terveydenhuollon puolella (Carbery ja Hegarty 2011).

Vertaisopetus ei ole ei ole mitenkään harvinainen tietotekniikan opetuksessakaan – päinvastoin, sitä on käytetty jo yli 20 vuotta auttamaan parantamaan opiskelijoiden suoritusta (Sperry ja Tedford 2008; Wills et al. 1994). Vertaisopetuksen käytön on todettu tuovan muun muassa seuraavia hyötyjä (Wills et al. 1994):

- Opiskelijat tutustuvat tarjolla olevaan materiaaliin paremmin ja he pohtivat aktiivisen opiskelun myötä asioita syvemmin.
- Opiskelijat tuntevat vahvempaa ryhmäytymistä.
- Yhteistyö lähentää opiskelijoita ja valmentaa heitä tulevaisuuden työelämään.

Vertaisoppiminen on siis opiskelijan kannalta hyväksi monessa suhteessa, mutta se auttaa myös opettajaa. Yksipuoleinen luennointi voidaan kokea tylsänä pakkopullana, mutta vertaisopetuksessa vaadittu opiskelijan aktiivisuus voi tuoda opettajallekin lisäintoa työhönsä.

Vertaisoppiminen ohjelmoinnissa

Ohjelmointi on perinteisesti nähty työnä, jossa tarvitaan keskittymistä – ns. flow-tila – jossa kaikki ylimääräinen ryhmäytyminen on pikemminkin haitaksi kuin hyödyksi. Vaikka tämä on osin edelleen täysin totta, ovat ketterät ohjelmistokehitysmenetelmät tuoneet mukanaan muun muassa pariohjelmoinnin (Begel ja Nagappan 2008). Tässä kyseisessä tekniikassa nimenomaisesti ohjelmointia tekee pari yhdessä yhden tietokoneen ääressä. Käytännössä toinen ohjelmoi ja toinen osallistuu tarkistamalla kirjoitettavaa koodia samanaikaisesti ja suunnittelemalla ohjelmaa eteenpäin. Tämä parantaa ohjelmakoodin laatua, vähentää virheitä ja levittää koodin ymmärrystä (Begel ja Nagappan 2008). Tässä siis selkeästi toteutuu vertaisoppiminen, kun ongelmaa ei ratko vain yksi ohjelmoija omassa siilossaan vaan kaksi yhdessä.

Pariohjelmointia on käytetty myös opetuksessa ja sen on havaittu parantavat ohjelmointitaitoja, opintojakson läpipääsyprosenttia, luottamusta omiin ohjelmointitaitoihin ja parempaa asennetta ohjelmointia ja tietotekniikkaa kohtaan ylipäätään (Braught, Wahls, ja Eby 2011). Tutkimustulos ei ole yllättävä, sillä yliopistoon tullessa opiskelijoiden ohjelmointitaidot vaihtelevat suuresti. Jotkut ovat voineen tehdä ohjelmointia jo työkseen, jolloin heille ensimmäiset ohjelmointikurssit ovat

lähinnä puuduttavan tylsiä, kun taas ensimmäistä kertaa ohjelmoinnin pariin pääsevät opiskelijat voivat kokea opintojakson hyvinkin vaikeaksi.

Oppimisen tukeminen

Toisena teoriana käytän Existence, Relatedness ja Growth (ERG, olemassaolo, liittyvyys ja kasvu) -teoriaa, joka pohjaa Maslown tarvehierarkiaan (Maslow 1943), mutta on sitä yksin kertaisempi ja täten helpompi ymmärtää ja toteuttaa (Schneider ja Alderfer 1973). Opintojakson sisältäessä monentasoista opiskelijoita opetuksen tulisi tukea eritasoista oppimista, tavoitteita ja intohimoja - tai niiden puutetta. Eri tasojen merkitykset on kuvattu taulukossa 4.

Taulukko 4. ERG-teoria ja sen sovittaminen ohjelmoinnin opetukseen.

	Teoria	**Ohjelmoinnin opetuksessa**
Existence	Yksilö tuntee turvallisuuden tunnetta, terveyttä ja saa ruokaa.	Esitellään opintojakson perustyökaluja - ja luodaan niistä YouTube-videot. Tarjotaan harjoitustila ja annetaan mahdollisuus kysyä ongelmakohdista sekä autetaan opiskelijan omaa ohjelmointia.
Relatedness	Yllä olevien lisäksi yksilöllä on mahdollisuus rakastaa ja tuntea kuuluvansa yhteisöön sekä ansaita kunnioitusta ja olla yhteisölle hyödyksi.	Kannustetaan opiskelijoita pariohjelmointiin ja opetetaan versionhallinnan käyttöä, jotta ryhmätyöskentely on sujuvampaa. Annetaan opiskelijalle mahdollisuus kerätä kunnioitusta virtuaalisessa oppimisympäristössä pisteiden ja edistymisen muodossa.
Growth	Yksilö voi olla luova ja täyttää potentiaalia, jota hänellä on.	Annetaan tarpeeksi vapaat kädet harjoitustyön suorittamiseen, jotta opiskelija voi halutessaan olla luova. Ohjeistetaan kuitenkin tarpeeksi tarkkaan, jotta eritasoiset opiskelijat eivät koe turhautumista.

ERG-teoria tarjoaa siis rungon rakentaa opintojakso alhaalta ylöspäin tarjoten opiskelijoille ensin perusasiat kuntoon ja kasaten tämän varaan syvempiä asioita. Tämä on siis verrattavissa reaalielämään, jossa Angry Birdsistä ei voi nauttia ennen kuin on katto pään päällä ja ruokaa vatsassa. Rakennettaessa opintojaksoa ERG-mallin avulla tulee asiat käytyä lävitse oikeassa järjestyksessä, eikä esimerkiksi harjoitustyön määrittelyä ruveta suunnittelemaan ennen kuin on selvyyttä millaisilla työkaluilla opiskelijat tulevat kurssisuoritteita rakentamaan. Samoin kurssin sisältö tulee peilata aiempien kurssien sisältöön, jotta päällekkäistä asiaa ei tule, mutta myöskään uusia asioita ei tuoda kurssille liikaa ja seuraavat opintojaksot saavat oikeanlaista hyötyä kehitettävästä opintojaksosta.

Oppimistilanne ei eroa tavallisesta työpaikasta siinä mielessä, että oppimisen hygieniatekijät (Herzberg 1968), kuten luokka, työvälineet, valaistus ja ilmastointi tulee olla kunnossa, jotta

opiskelija voi motivoitua omaan tekemiseensä. Ohjelmoinnin opetuksessa hygieniatekijöihin voidaan lukea myös ohjelmointiympäristö työkaluineen ja ulkopuolisine ohjeineen.

Rakentamalla opintojakso ERG-mallin mukaisesti, tarjotaan kaikille opiskelijoille tarvittavat perusfasiliteetit, ohjeistus niiden käyttöön ja apua ilmeneviin ongelmakohtiin. Luodaan tilanne, jossa opiskelija tuntee ympäristön turvalliseksi. Tämän lisäksi kannustetaan opiskelijoita tekemään harjoitustyötä parityönä kuitenkaan pakottamatta siihen. Harjoitustyötä ei kuitenkaan sidota pilkuntarkasti, joten opiskelija saa olla luova, ja mikäli hän haluaa ylittää omat rajansa ja näyttää opettajalle, että ”täältä pesee”, on siihen mahdollisuus.

Tulevaisuuden kehitysehdotukset

Tähänastiset tulokset

Osa uudistuksista tuli opintojaksolle jo syksyllä 2013, mutta suurimmat muutokset on tarkoitus tuoda vuoden 2014 opintojakson suurempaan kehitykseen. Opintojakso on rakentunut viimeisen kymmenen vuoden ajan niin verkko kuin lähiopetuksen varaan (sulautuva oppiminen), mutta nyt painopistettä siirrettäneen vielä entistä enemmän verkkoon. Opintojakson palaute on ollut hyvää koko sen ajan, mitä olen sitä vastuullisesti opettanut, ja opiskelijat ovat arvostaneet niin nauhoitettuja luentoja kuin vapaamuotoista harjoitustyötä. Parannettavaa on kuitenkin edelleen jäänyt.

Kehitysajatukset syksylle 2014

Yliopiston siirtyessä kuuden viikon periodeihin syksyllä 2014, tulee myös tämän opintojakson viikkomäärä tippumaan 14:stä 12:een. Tämä aiheuttaa tiettyjä muutoksia. Samalla ohjelmointikieli vaihtunee C++:sta Javaan, joten muutoksia jouduttaneen tekemään kesän 2014 aikana suuresti. Seuraavassa on yhteenveto tulevista muutoksista.

Ohjelmointikieli vaihtuu C++:sta Javaan. Yliopistomme on pysynyt C++:n opetuksessa mukana lähinnä Nokian Symbian-käyttöjärjestelmän takia, mutta nyt sekä Nokian puhelinbisnes, Symbian, että kolmannen osapuolen ohjelmisto-osaaminen on jäänyt historiaan, joten tarve C++-osaajille on vähentynyt tällä teollisuuden alalla. C++ ei kielenä pakota opiskelijaa oliomaiseen ohjelmanrakentamiseen, koska se on ei ole puhdas oliokieli, vaan hybridikieli, jolla voi ohjelmoida niin proseduraalisesi, oliomaisesti kuin funktionaalisestikin. Vaihto Javaan tuo käyttöön kielen, joka on lähes puhtaasti oliokieli ja pakottaa opiskelijan rakentamaan oliopohjaisia ratkaisuja proseduraalisten sijaan.

Tähän mennessä opintojakson harjoitustyön on voinut tehdä tekstipohjaisella käyttöliittymällä, koska opintojakson vaatimuksiin ei ole kuulunut graafisen käyttöliittymän opetusta saati sen

käyttöä harjoitustyössä. Graafisen käyttöliittymän tekemistä on kuitenkin lyhyesti esitelty luennolla ja sen käytöstä harjoitustyössä on saanut enemmän pisteitä. Java on suunniteltu jo suoraan tukemaan graafisia käyttöliittymiä ja tarkoitus on ottaa käyttöön jokin avoimen datan palvelun ohjelmointirajapinta ja rakentaa harjoitustyö käyttämään tämän palvelun dataa jollain mielekkäällä tavalla. Palvelun käyttö tulisi olla graafisen käyttöliittymän lävitse. Tällä erää pyörittelemme ajatuksia kartalla liikkuvista SmartPost-paketeista ja Skinnarilan pizzapalveluista.

Nykyisellään opintojakson luentonauhoitteet ovat koostuneet 40 – 90 minuuttisista luennoista, joilla käsitellään yhdestä useampaan teemaa. Luentojen tarjoaminen YouTubessa on saanut kehuja, mutta on tullut myös palautetta, että kokonaisuudet ovat liian pitkiä, jolloin keskittyminen herpaantuu. Koska luennoilla käy hyvin vähän opiskelijoita ja vuorovaikutuksen määrä on vähäinen, ei tulevaisuudessa nähdä tarpeelliseksi järjestää erillisiä luentoja, vaan ainoastaan YouTubessa sijaitsevia videokokonaisuuksia, jotka on jaoteltu asiakokonaisuuksiin pitkien luentojen sijaan. Interaktiivisuus pyritään järjestämään harjoituksissa.

Harjoitusten roolia tullaan muokkaamaan enemmän opiskelijoita kerääväksi porrastamalla niiden vaikeusastetta, liittämällä ne tiiviimmin kiinni viikon YouTube-videoissa käsiteltyihin asioihin ja tarjoamalla harjoitusten tekemisestä pisteitä opintojaksosuoritukseen. Harjoitukset ovat jatkossa ensisijainen paikka, jossa opiskelija voi kysyä opintojakson vastuuhenkilöltä ratkaisuja ongelmiinsa. Mikäli tarvetta ilmenee, voidaan harjoitusten pituutta kasvattaa kahdesta kolmeen tuntiin. Harjoituksissa opiskelijat voivat ohjelmoida yksin tai parina niin halutessaan. En koe tarpeelliseksi pakottaa opiskelijoita tekemään töitä toisen opiskelijan kanssa mikäli he eivät niin halua. Kannustan kuitenkin heitä tekemään muun muassa harjoitustyön parityönä. Mikäli kaikki kurssille ilmoittautuvat opiskelijat tulisivat harjoituksiin, tilat pakottaisivat pariohjelmointiin, mutta ei liene realistista olettaa kaikkien saapuvan harjoituksiin, vaikka niistä jaettaisiin pisteitä kurssisuoritukseen.

Tähän mennessä luentokalvot on tarjottu Nopassa pdf-muodossa ja luentoesimerkit ladattavina kooditiedostoina. Jatkossa teoriaosuus on tarkoitus tarjota Google-dokumenttina, jolloin opiskelija pääsee siihen käsiksi niin tietokoneella, tabletilla kuin puhelimellakin. Teoriaosuus tarjoaa perinteiset luentokalvot, jotka on käsitelty luentovideoilla, mutta myös kaksi Java-opasta, joista toinen keskittyy Java-kieleen ja toinen muihin kurssilla käsiteltyihin asioihin, kuten graafiseen käyttöliittymään, avoimeen dataan ja karttoihin. Opiskelijoille tarjoutuu myös suora mahdollisuus jättää kommentteja materiaaliin, jolloin virheet löytyvät ja korjaantuvat nopeammin. Samoin esimerkkiohjelmakoodit tarjotaan suoraan versionhallinnan kautta GitHubissa tai vastaavassa, jolloin opiskelija saa kloonattua ne itselleen yhdellä komennolla ja mikäli virheitä löytyy voi hän puskea korjausehdotuspyynnön takaisin varastoon.

Yliopistomme otettua Moodlen käyttöön on tentti mahdollista siirtää digitaaliseksi ja päästä eroon paperille ohjelmoinnista. Tentti voidaan toteuttaa niin, että opiskelija voi suorittaa sen itselleen parhaiten sopivana aikana, esimerkiksi, kahden päivän aikavälillä. Opiskelijoille voidaan myös arpoa eri kysymyksiä, joten tentin tekeminen ryhmätyönä vaikeutuu, mikäli koetaan, ettei se ole suotavaa.

Koska opiskelijat ovat erilaisia ja ovat valmiita panostamaan opintojakson eteen eritasoisesti, ei nähdä pakolliseksi olettaa, että kaikki opiskelijat suuntaisivat ERG-mallin kolmannelle growth-tasolle. Joillekin opintojakso voi olla vain täysin pakollinen, jotta tutkinnon saa kasaan, jolloin hänelle riittää, että existence-taso täytyy, eli opintojaksolla on paikka rakentaa ohjelmakoodia, YouTubesta löytyvät opetusvideot ja muu materiaali on helposti tarjolla sekä assistentilta saa apua tarvittaessa. Toivottavissa on, että mahdollisimman moni kiinnostuu asiasta ja haluaa rakentaa harjoitustyötä parityönä, jolloin ryhmätyöskentelytyökalut, kuten versionhallinta, tulevat ajankohtaisiksi.

Tarkoitus on siis tällä opintojaksolla teknillisesti erottaa muuttuvat asiat muuttumattomista ja tarjota muuttumattomat asiat, eli luennot, videokokonaisuuksina YouTubessa ja poistaa fyysiset luennot kokonaan. Sitä vastoin panostusta siirretään muuttuviin asioihin, eli tässä tapauksessa harjoituksiin ja harjoitustyöhön.

Johtopäätökset

Opintojakson yhden osan kehittäminen heijastuu myös muihin osiin ja nyt yliopiston päätettyä lyhentää periodeja viikolla ja koulutusohjelman päätettyä vaihtaa kurssin opetuskieli C++:sta Javaan tarjoutui hyvä mahdollisuus uusia koko kurssi kerralla. Opiskelijapalaute on ollut kurssilla hyvää, mutta se ei tarkoita, etteikö parantamisen varaa olisi.

Opintojaksolla päätettiin luopua luentomuotoisesta lähiopetuksesta ja korvata se verkossa olevalla luettavilla oppimateriaaleilla ja opetusvideoilla. Lähiopetuksen painopiste siirrettiin harjoituksiin, joihin opiskelijoiden osallistumista lisätään antamalla suoritetuista harjoituksista pisteitä.

Tulevaisuudessa harjoitusten tekemistä virtuaalisessa oppimisympäristössä tullaan pohtimaan mahdollisena tehtävien palautuskanavana, mikäli se tuntuu järkevältä, eikä vähennä radikaalisti opiskelijoiden intoa osallistua harjoituksiin.

Itselle tämä kehityshanke yhdessä koko yliopistopedagogiikan opintojakson kanssa avasi mielen monille ajatuksille. Ajatukset sekä luentojen siirtämisestä täysin verkkoon että paperisen tentin eliminoinnista ovat syntyneet tämän kurssin uusimisprosessin aikana. Yliopistopedagogiikka on saanut kyseenalaistamaan päähäni pinttyneet ajatukset ja korvannut ”musta tuntuu” mielipiteet

paremmin tieteellisesti tuetuilla tutkimustuloksilla.

Lähteet

Begel, Andrew, and Nachiappan Nagappan. 2008. “Pair Programming: What’s in It for Me?” In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 120–128. Kaiserslautern, Germany: ACM.

Boud, David, and Alison Lee. 2005. “‘Peer Learning’ as Pedagogic Discourse for Research Education.” *Studies in Higher Education* 30 (5): 501–516. doi:10.1080/03075070500249138.

Braught, Grant, Tim Wahls, and L. Marlin Eby. 2011. “The Case for Pair Programming in the Computer Science Classroom.” *Trans. Comput. Educ.* 11 (1): 1–21.

Carbery, Alan, and Nora Hegarty. 2011. “Introducing Problem-Based Learning into One-Shot Information Literacy Instruction at Waterford Institute of Technology Libraries.” In *SCONUL Focus*, 53:30–33. Ireland.

Herzberg, Frederick. 1968. “One More Time: How Do You Motivate Employees?” *Harward Business Review* 46 (1): 53–62.

Lewis, John. 2000. “Myths about Object-Orientation and Its Pedagogy.” *ACM SIGCSE Bulletin* 32 (1): 245–249. doi:10.1145/331795.331863.

Maslow, Abraham. 1943. “A Theory of Human Motivation.” *Psychological Review* 50: 370–396.

Moskal, Patsy, Charles Dziuban, and Joel Hartman. 2013. “Blended Learning: A Dangerous Idea?” *The Internet and Higher Education* 18 (July): 15–23. doi:10.1016/j.iheduc.2012.12.001.

Schneider, Benjamin, and Clayton P Alderfer. 1973. “Three Studies of Measures of Need Satisfaction in Organizations.” *Administrative Science Quarterly* 18 (4): 489–505.

Sperry, Rita A., and Phyllis Tedford. 2008. “Implementing Peer-Led Team Learning in Introductory Computer Science Courses.” *J. Comput. Sci. Coll.* 23 (6): 30–35.

Vilner, Tamar, Ela Zur, and Judith Gal-Ezer. 2007. “Fundamental Concepts of CS1: Procedural vs. Object Oriented Paradigm - a Case Study.” *ACM SIGCSE Bulletin* 39 (3): 171. doi:10.1145/1269900.1268835.

Wai, Cho Cho, and Ernest Lim Kok Seng. 2014. “Exploring the Effectiveness and Efficiency of Blended Learning Tools in a School of Business.” *Procedia - Social and Behavioral Sciences* 123 (March): 470–476. doi:10.1016/j.sbspro.2014.01.1446.

Wills, Craig E., David Finkel, Michael A. Gennert, and Matthew O. Ward. 1994. “Peer Learning in an Introductory Computer Science Course.” *SIGCSE Bull.* 26 (1): 309–313.