OHJELMOINNIN OPETTAMINEN EMPIRICA CONTROLILLA

Harri Kähkönen ja Kai Piiroinen

24.5.2002

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

ESIPUHE

Pro gradu -tutkielmamme aihe muodostui vähitellen useiden kuukausien aikana. Idea tutkia ohjelmoinnin opettamista Empirica Control -ohjelmistolla sai alkunsa tietokoneavusteisen opetuksen projektityöstä, jossa suurta roolia näytteli Empirica-mittausjärjestelmä. Projekti toteutettiin kolmihenkisenä, mukanamme oli kemian opiskelija. Lopullisen muotonsa tutkimusongelmamme sai professori Erkki Sutisen ideoimana.

Tutkielmaa tehdessämme olemme kohdanneet monia hidasteita, jotka osaltaan ovat muokanneet tutkielman lopullista muotoa. Suurimmat ongelmat kohtasimme opetuskokeilun järjestelyissä: aikataulujen yhteensovittaminen oli hankalaa. Lieksan lukion ja Merilän perusasteen rehtoreiden myötämielisellä tuella saimme järjestettyä opetuskokeilut tammi–helmikuussa 2002. Lieksan lukiolla suoritimme opetuskokeilun Kai Piiroisen johdolla tammikuussa. Tarkoituksenamme oli tutkia, kuinka Empirica Control -ohjelmisto soveltuu ohjelmoinnin perusrakenteiden opettamiseen. Tahdomme kiittää lukion rehtoria Pekka Tolvasta sekä tietotekniikanopettaja Markku Säilyä. Heidän avustuksellaan saimme käyttöömme tilat ja työrauhan sekä kahdeksan lukion ensimmäisen luokan oppilasta mukaan kokeiluun. Merilän perusasteella opetuskokeilu tapahtui Harri Kähkösen johdolla helmikuussa kahtena perättäisenä keskiviikkona. Tutkimustavoitteet olivat samat kuin lukiossa. Opetuskokeiluun osallistui kahdeksan viidennen ja kuudennen luokan oppilasta. Perusasteen rehtori Arto Turpeinen ansaitsee suuret kiitokset tilojen ja työrauhan järjestämisestä. Kiitokset myös kokeiluun osallistuneille oppilaille innostuksesta ja mielenkiinnosta.

Totesimme jo tutkimuksemme alkuvaiheessa, että molemmat opetuskokeilut tarjoavat yhdessä paremmat ja kattavammat mahdollisuudet aineiston analysointiin ja tulosten saavuttamiseen. Tämän vuoksi emme halunneet hajottaa tutkielmaamme kahteen erilliseen osaan. Erottelemme tutkimuksemme lähinnä opetuskokeilujen osalta, vaikka osallistuimme molemmat niihinkin aktiivisesti havainnoiden ja ohjaten toimintaa. Itse tutkimuksen raportointi oli alusta loppuun yhteistyötä. Näin tiivis yhteistyö oli mahdollista, koska opinnot etenivät molemmilla suunnilleen saman aikataulun mukaisesti.

Suuret kiitokset kuuluvat Joensuun yliopiston professori Erkki Sutiselle ja professori Veijo Meisalolle Helsingin yliopistosta, jotka toimivat pro gradumme ohjaajina ja tarkastajina. Professori Sutinen toimi pro gradu -tutkielmamme ohjaajana yliopistollamme Joensuussa ja pro-

fessori Meisalo toisena tutkimuksemme tarkastajana. Professori Sutinen antoi tarvittaessa tukea tutkielman edetessä sekä vauhditti raportointiprosessia. Erityiset kiitokset professori Meisalolle tuesta, jonka saimme sähköpostitse useina kiperinä hetkinä. Professori Markku Tukiainen Joensuun yliopistolta tarjosi meille apua varsinkin visualisointiin sekä ohjelmoinnin oppimiseen ja opettamiseen liittyvissä kysymyksissä. Hän suositteli meille monia hyödyllisiä artikkeleita ja opasti opetuskokeilun suorittamiseen liittyvissä kysymyksissä. Joensuun yliopiston suunnittelija Jari Kukkonen auttoi keräämämme aineiston analysointia koskevissa kysymyksissä. Empirica-mittausjärjestelmään liittyvissä asioissa saimme tukea Helsingin yliopiston professori Jari Lavoselta. Lisäksi haluamme kiittää Joensuun normaalikoulua, sillä koulu mahdollisti osaltaan opetuskokeilun toteuttamisen lainaamalla käyttöömme Empirica-mittausjärjestelmän ja useita antureita.

Suurimmat kiitokset tutkielmamme valmistumisesta kuuluvat kuitenkin vanhemmillemme ja rakkaillemme.

Joensuussa 24.5.2002,
Harri Kähkönen & Kai Piiroinen

TIIVISTELMÄ

Tutkielmamme pohjautuu opetuskokeiluun, jossa tutkittiin Empirica Control -ohjelmiston soveltuvuutta ohjelmoinnin perusrakenteiden opettamiseen. Empirica Control on visuaalinen ohjelmointiympäristö, jota voidaan hyödyntää osana Empirica-mittausjärjestelmää, jolloin toiminta voidaan sitoa konkreettiseen maailmaan. Opetuskokeilut suoritettiin Lieksassa tammi–helmikuussa 2002 lukiolla sekä perusasteella. Lieksan lukiolla kokeiluun osallistui kahdeksan ensimmäisen luokan oppilasta. Merilän perusasteella järjestettyyn kokeiluun osallistui kahdeksan viidennen ja kuudennen luokan oppilasta. Oppilaat toimivat kokeilussa neljän hengen ryhmissä ratkaisten itse määrittelemiään ongelmia Empirica-mittausjärjestelmän ja Empirica Control -ohjelmiston avulla. Tutkimuksemme perusteella voidaan todeta, että Empirica Controlin käyttö opetuksessa kannustaa oppilaita tutkivaan, luovaa ongelmanratkaisua hyödyntävään oppimiseen. Empirica Controlin avulla ohjelmoinnin perusrakenteiden opettamisesta ja oppimisesta tulee helpompaa ja mielekkäämpää kuin perinteisten ohjelmointikielten avulla.

Avainsanat: Empirica Control, ohjelmoinnin opettaminen, konstruktivismi, visuaalinen ohjelmointi

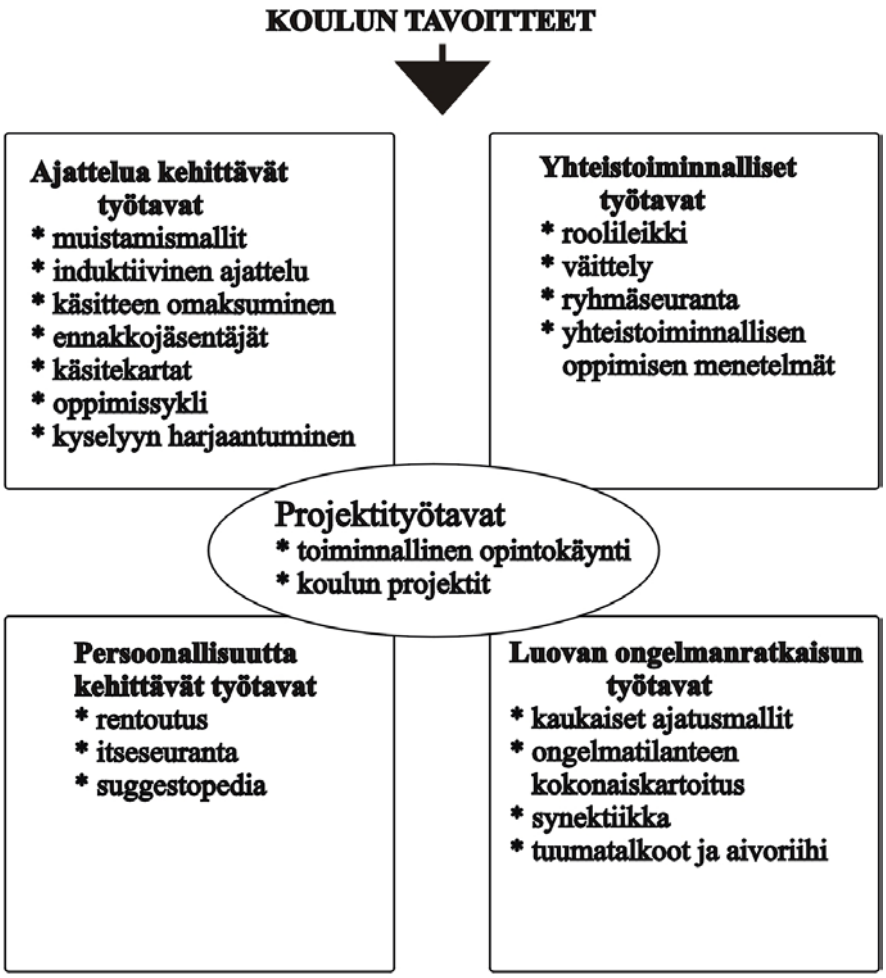
1 JOHDANTO

Tässä tutkimuksessa osoitamme, kuinka ikonipohjainen, konkretisoiva ohjelmointi edistää oppilaiden käsitystä ohjelmoinnin perusrakenteista (silmukka, ehto, muuttuja) sekä madaltaa kynnystä ohjelmointiin. Välineenä käytämme Empirica-mittausjärjestelmää, joka on Windows-ympäristössä toimiva luonnontieteiden ja teknologian opetukseen ja opiskelemiseen soveltuva järjestelmä (Lavonen et al., 1996). Osoitamme kuinka graafisen, konkretisoivan järjestelmän käyttäminen opetuksessa motivoi oppilaita sekä mahdollistaa uudenlaisten oppimisprosessien ja ongelmanratkaisutapojen syntymisen. Näin ollen ratkaistavasta ongelmasta voidaan Empirica-mittausjärjestelmän avulla rakentaa konkreettinen esitys, jonka kautta oppilas voi ratkaista ongelman muokkaamalla fyysistä ympäristöään haluamallaan tavalla.

Tutkimuksen perustana toimii Papertin (1980) ajatus. Hänen mukaansa lapset toimivat tietoteoreetikkoina ohjelmoidessaan koneita. Ohjelmoidessaan LOGO-ympäristöllä lapset oppivat siis tietoa muokkaamalla ja tutkimalla. Papert tutki LOGO-ohjelmoinnin vaikutusta matematiikan opetuksessa ja havaitsi, että oppimisprosessi muuttuu lapsen oppiessa ohjelmoimaan – hän hankkii tietoa itse tunnistettuihin tarkoituksiin. Ohjelmoinnilla saavutetaan uusi ajattelutapa, koska perinteinen oikein-väärin-ajattelu muuttuu virheen korjaamista ja ongelmanratkaisuprosesseja hyödyntävään ajatteluun.

Perinteisessä ohjelmoinnin opettamisessa on turvauduttu opetusteknologiseen lähestymistapaan. Opetusteknologinen lähestymistapa pohjautuu behavioristiseen oppimiskäsitykseen. Tässä oppimiskäsityksessä pyritään systemaattiseen, tarkkaan määriteltyjen opetustavoitteiden saavuttamiseen. Ihminen on kohde, jonka käyttäytymiseen voidaan vaikuttaa ympäristöä säätelemällä (Vaherva, 1986). Sahlberg ja Leppilampi (1994) kuvaavat koulun tavoitteiden vaikutuksia käytettäviin opetusmenetelmiin (kuva 1). Perinteisessä ohjelmoinnin opetuksessa opetusmenetelmät eivät kuitenkaan toteudu heidän kuvaamallaan tavalla. Opetuskokeilun aikana oppilaat toimivat jatkuvasti ryhmissä, jolloin yhteistoiminnalliset työtavat kuten väittelyt, keskustelut ja ideoinnit ohjasivat ryhmien toimintaa. Ohjelmoinnin perusrakenteita opetettaessa ryhmät muodostivat muistamismalleja: opetettavat asiat pyrittiin liittämään todelliseen maailmaan esimerkkien muodossa. Luovaa ongelmanratkaisua ryhmät hyödynsivät määrittelemänsä ongelman ratkaisussa. Ryhmien toimintaa tehostettiin esittämällä kysymyksiä laaditun ohjelman toiminnasta, jolloin ryhmät joutuivat pohtimaan ohjelman toimintaa

sekä ryhmän tavoitteiden toteutumista. Opetuskokeilun aikana Sahlbergin ja Leppilammin määrittelemät opetusmenetelmien tavoitteet siis toteutuivat, toisin kuin perinteisessä ohjelmoinnin opettamisessa.



Kuva 1: Koulun tavoitteiden vaikutus opetusmenetelmiin (kts. Sahlberg & Leppilampi, 1994).

Tutkimuksemme antaa viitteitä siitä, kuinka tietotekniikan integroimisella kouluopetukseen luodaan edellytyksiä modernin yhteiskunnan vaatimaan elinikäiseen oppimiseen. Tavoitteenamme on selvittää kuinka Empirica Controlin avulla voidaan tukea oppilaiden aktiivista oppimisprosessia sekä ohjelmoinnin opetusta. Selvitämme myös Empirica Controlin vaikutukset oppilaiden asenteisiin ohjelmointia ja tietotekniikkaa kohtaan. Tarkastelemme myös sitä, kuinka ohjelmointia tulisi hyödyntää opetuksessa, kun otetaan huomioon tietoyhteiskunnan koululaitoksille asettamat vaatimukset. Tutkimuksemme keskeisin tarkoitus on selvittää, kuinka Empirica Controlin avulla voidaan opettaa ohjelmoinnin perusrakenteita kuten ehtolausetta, silmukkaa sekä muuttujia.

Papert (1980) tutki LOGO-ohjelmointikielen vaikutusta lasten matematiikan oppimisessa. Hän havaitsi oppimisen luonteessa muutoksia, kun lapset ryhtyivät itse hankkimaan tietoa ja toimivat aktiivisina oppijoina. Hänen mukaansa lapset kykenevät ratkaisemaan ongelmia ja tehtäviä oppimisympäristössä joka tarjoaa riittävästi älyllistä ja henkistä tukea. Papertin kehittämään LOGO-ohjelmointikieleen pohjautuu LEGO/LOGO-ohjelmointikieli. Järvinen (1998) käytti LEGO/LOGO-ohjelmointikieltä tavoitteenaan perehdyttää perusasteikäisille lapsille ohjelmointitaitoja sekä modernin teknologian mahdollisuuksia. Empirica Control -ohjelmiston avulla voidaan saavuttaa samankaltaisia tuloksia opetuksessa. Lund (1999) toteaa Lego-robottien avulla tekemässään tutkimuksessa, että fyysisten olioiden kanssa toimiminen on tärkeää, jotta lapset voivat oppia todellisesta maailmasta. Robottien avulla toimimisen kautta lapset saavuttavat Papertin kuvaaman konstruktivistisen oppimismallin tavoitteet (Lund, 1999; Miglino et al., 1999).

Tietotekniikan opetusta voidaan laajentaa integroimalla sitä muihin oppiaineisiin. Kocijancic (2001) toteaa teknologian tieteidenvälisen ja integroidun opetuksen lisäävän oppilaiden kiinnostusta teknologiaa kohtaan. Tämä edellyttää kuitenkin opettajilta taitoa ja kykyä soveltaa teknologiaa muiden aineiden opetukseen. Kolberg ja Orlev (2001) tutkivat teknologian integroimista perusasteikäisten oppilaiden koulutukseen. Heidän alustavien tulostensa mukaan oppilaiden toiminta on huomattavasti oletettua nopeampaa sekä ryhmien työskentely aktiivisempaa, kun teknologiaa hyödynnetään opetuksessa. Opetuskokeilumme tukee näitä havaintoja – oppilaat työskentelivät Empirica Control -ohjelmiston avulla innokkaasti ja ryhmien laatimat ohjelmat olivat monimutkaisempia kuin oletimme.

Opetuskokeilun aikana oppilaat toimivat ryhmissä ratkaisten itse määrittelemiään ongelmia. Sutinen ja Tarhio (2001) keskittyvät tutkimuksessaan ongelmanratkaisuprosessin luovaan tai innovatiiviseen ongelmanhallintaan. Heidän mukaansa luovan ongelmanratkaisuprosessin edellytys on ongelmien sekä mahdollisten ratkaisujen oikeanlainen hallinnointi. Luovan ongelmanratkaisuprosessin tulisi olla avoin jatkuville muutoksille.

Luvussa kaksi tutustumme ohjelmoinnin opettamiseen, ongelmanratkaisuun sekä *konstruktivistiseen* oppimiskäsitykseen Papertin (1980) ja Ben-Arin (2001) ajatusten pohjalta. Tutustumme myös visualisoinnin hyödyntämiseen ohjelmoinnin opetuksessa (luku 3). Käsittelemme varsinkin ohjelmoinnin visualisointia, visuaalista ohjelmointia sekä konkreettista ohjelmointia. Luvussa neljä käsittelemme Empirica-mittausjärjestelmän hyödyntämistä opetuk-

sessa sekä tutustumme avoimen oppimisympäristön määritelmään. Luvuissa viisi ja kuusi käymme läpi tutkimusasetelman (luku 5) sekä tutkimuksessa käytetyt metodit (luku 6). Opetuskokeiluja, jotka suoritettiin Lieksassa perusasteella ja lukiossa käsitellään luvussa seitsemän. Tutkimuksen tulokset ja niiden analysointi esitellään luvussa kahdeksan. Luku yhdeksän sisältää yhteenvedon sekä ehdotuksia jatkotutkimuksia varten.

Ohjelmoinnin opettamista ja visuaalisuutta ohjelmoinnin oppimisen tukena käsitteleviä lukuja sekä Empirica Control -ohjelmistoa yhdistävät tekijät on kuvattu taulukossa 1. Empirica Controlissa yhdistyvät ohjelmoinnin oppimisessa sekä visuaalisuuden hyödyntämisessä näkemämme edut taulukossa esitetyllä tavalla. Taulukon sarakkeilla käsitellään luvussa 2 esitettyjä ohjelmoinnin opettamisessa käytettävän ohjelmointikielen ominaisuuksia. Riveille kokoamme visualisoinnin ohjelmoinnin oppimista edistäviä piirteitä (luku 3).

Visuaalisuus ohjelmoinnin oppimisen tukena \ Ohjelmoinnin opettaminen	Yksinkertaisuus (Ohjelmointikielen opittavuus, käytettävyys)	Ilmaisuvoima (Kuinka luonnollista käyttää kieltä ongelmanratkaisuun)	Yhteensopivuus (Kielen sitominen uusien sovellusalueiden opetukseen)	Kognitiivinen rikkaus (Kuinka hyvin kieli tukee oppilaiden ajatusmaailmaa)	Ongelmanratkaisu (Kuinka kieli tukee ongelmanratkaisuprosessia)
Visuaalinen ohjelmointi	Kuvakkeet selkeitä, ohjelma rakentuu automaattisesti (syntaksi oikein).	Empirica Control tukee luonnontiet. ongelmien ratkaisua. Muuten rajoitteita.			Ulkoa muistamisen vähäinen tarve vapauttaa ongelmanratkaisuun enemmän resursseja.
Ohjelman suorituksen animointi	Sininen pallo kuvaa ohjelman suorituksen etenemisen			Ohjelman suorituksen animointi hyödyntää todellisen maailman tilanteita (liikennevalot).	Ohjelman suoritusta seuraamalla voidaan tutkia mikä ratkaisussa menee pieleen.
Ohjelmoinnin konkretisointi	Ohjelmointi voidaan nähdä sinisen pallon ohjaamisena.			Ohjelmointi sinisen pallon ohjaamista.	Ongelmaa ratkaistaessa voidaan sinistä palloa ohjata ratkaisevaan suoritukseen.
Järjestelmän käytettävyys	Loogisten operaatioiden muodostaminen ja rakenteiden muokkaaminen hankalaa.		Empirica Controlissa yhteensopivuus on liian rajallinen.		

Taulukko 1: Empirica Controlin ominaisuudet ohjelmoinnin oppimisen tukena.

2 OHJELMOINNIN OPETTAMINEN *(Kai Piiroinen)*

Ohjelmoinnin oppiminen ei suinkaan ole helppoa. Oppilaiden kyky muodostaa käyttökelpoinen *mentaalimalli* tietokoneen ja ohjelmien toiminnasta on usein puutteellista, mikä vaikeuttaa ohjelmoinnin oppimista. Ben-Arin (2001) mukaan varsinkin aloittelevalla ohjelmoijalla voi olla suuria hankaluuksia hahmottaa ohjelman toimintaa. Ohjelmoinnin opettaminen *Empirica Control* -ohjelmiston avulla antaa tehokkaan mahdollisuuden konstruktivistisen opetussuuntauksen hyödyntämiseen. Opetuksen ja oppimisen tutkimusta hallitsee nykyisin pitkälti *konstruktivistinen oppimiskäsitys* (Ben-Ari, 2001). Tässä luvussa käymme läpi konstruktivismin keskeisiä periaatteita sekä kyseisen oppimiskäsityksen soveltuvuutta ohjelmoinnin opettamiseen. Lisäksi tutustumme ongelmanratkaisuun oppimisprosessin osana.

2.1 Konstruktivistinen oppimiskäsitys

Konstruktivistisen oppimiskäsityksen mukaan oppiminen on yksilön aktiivisen *tiedon konstruoimisprosessin* tulos (Ben-Ari, 2001). Tästä seuraa, että jokaisen yksilön aiemmat tiedot ovat vaikuttamassa uuden tiedon syntyyn – vanhaa tietoa pikemminkin muokataan sopivaksi kuin uutta tietoa sisäistetään. Tiedon omaksuminen ei siten ole passiivinen prosessi. Konstruktivistisen oppimiskäsityksen mukaan on tärkeää, että jokainen yksilö omaksuu opittavasta asiasta yksilöllisen kognitiivisen rakenteensa. Ben-Arin mukaan tämä rakenne ei välttämättä vastaa tieteellisesti todennettavissa olevaa käsitystä, jolloin se tulee oikaista oppimisprosessin aikana. Konstruktivistinen oppimiskäsitys johtaa Rauste-von Wrightin (1997) mukaan joustavaan sekä oppijan valmiudet huomioivaan opetuksen korostamiseen.

Ben-Ari (2001) toteaa, että oppilaan muodostamaan tietämykseen vaikuttavat hänen aikaisemmat kokemuksensa opetettavasta asiasta. Oppimisprosessiin vaikuttavat myös oppilaan omaksuma oppimistyyli sekä henkilökohtaiset luonteenpiirteet. Rauste-von Wrightin (1997) mukaan oppijan tapa hahmottaa maailmaa ja sen tulkintaan käytettyjen käsitteiden tulisi olla opetuksen lähtökohtana. Hänen mukaansa koulumaailmassa tulisi painottaa tiedon ymmärtämistä ulkoaoppimisen sijaan, mikä edistäisi tiedon mielekästä konstruointia. Konstruktivistisen oppimiskäsityksen mukainen opettaminen on Ben-Arin mukaan huomattavasti vaativampaa kuin klassisen opettamistyylin mukainen opetus. Opettaja ei olekaan enää aktiivinen tiedon jakaja ja oppilas passiivinen vastaanottaja. Konstruktivistisessa oppimiskäsityksessä opettaja ja oppilas saavuttavat tasavertaisemman aseman oppimistilanteessa. Papert (1980) havaitsi

tutkiessaan LOGO-ympäristön käyttöä opetuksessa samankaltaisia vaikutuksia – lapsi oppii, että opettajakin on oppija ja jokainen oppii virheistään. Papert kritisoikin koulujärjestelmän tapaa hylätä oppilaan väärät teoriat, jolloin samalla hylätään tapa, jolla lapset todella oppivat. Aktiivisessa oppimisprosessissa opettajasta tulee ohjaaja, joka omalla panoksellaan ohjaa oppimista oikeaan suuntaan.

Ben-Arin (2001) mukaan passiivinen oppiminen, jossa oppilas toimii pelkkänä tiedon vastaanottajana todennäköisesti epäonnistuu. Jokainen oppilas jäsentää uuden tiedon omalla tavallaan, koska opettaja pelkkänä tiedonjakajana ei kykene tarjoamaan tietoa yksilöllisesti. Hänen mukaansa oppimisen tulisi siten olla aktiivista, jolloin oppilas voi konstruoida tietoa yksilöllisesti opettajan ohjauksen alaisuudessa saaden samalla palautetta muilta oppilailta. Enkenberg (1998) toteaa, että konstruktivistinen tulkinta itse oppimisesta sitoo oppimisen oppilaan itse luomiin teorioihin ja käsityksiin.

Konstruktivistit uskovat, että tehokas oppiminen ei pelkästään vaadi tosiasioiden löytämistä vaan oikeellisten rakenteiden konstruointia. Konstruktivistisen oppimistilanteen saavuttamiseksi vaaditaankin opettajien aktiivista ohjausta ja ohjauksen tulee perustua käsitykseen oppilaan senhetkisestä tiedon *kognitiivisesta* rakenteesta (Ben-Ari, 2001). Rauste-von Wrightin (1998) mukaan kouluissa ei ole vielä kyetty tarjoamaan *oppimisympäristöjä*, jotka ehkäisisivät yhteiskunnasta vieraantumista. Hänen mukaansa tällainen oppimisympäristö mahdollistaisi tarvittavien perusvalmiuksien ja ongelmanratkaisutaitojen hallinnan sekä eettisten ratkaisujen tekemisen kehittymiseen. Empirica Control tarjoaa mahdollisuuden laajemman oppimisympäristön kehittämiseen, jossa Rauste-von Wrightin oppimisympäristöille esittämät vaatimukset voidaan täyttää. Empirica Controlin avulla voidaan oppilaita aktivoida kehittämään uusia ongelmia ratkaistaviksi. Koulut ovat yleisesti valmistaneet oppilaita ratkomaan eilispäivän ongelmia, huomioimatta tulevaisuuden mukanaan tuomia haasteita (Rauste-von Wright, 1998).

Ben-Arin (2001) mukaan konstruktivistit suosivat moderneja *opetusmetodeja*, kuten ryhmätyöskentelyä, tutkivaa oppimista ja aktiivista toimintaa vain silloin, kun niillä pyritään auttamaan oppilaita konstruoimaan toteutuskelpoisia tiedon rakenteita. Toteutuskelpoisten rakenteiden tulisi pohjautua aiemmin opittuun tietoon. Hän toteaakin konstruktivismilla ja *tutkivalla oppimisella* olevan paljon yhteistä. Brunerin (1962) mukaan tutkivan oppimisen hyötyjä ovat (1) älyllisen ja taitavan ajattelun potentiaalin lisääntyminen, (2) huomion kään-

tyminen ulkoisista palkinnoista sisäisiin, (3) löytämisen heuristiikan oppiminen ja (4) muistin säästäminen.

2.2 Konstruktivismi ja tietojenkäsittelytieteen opetus

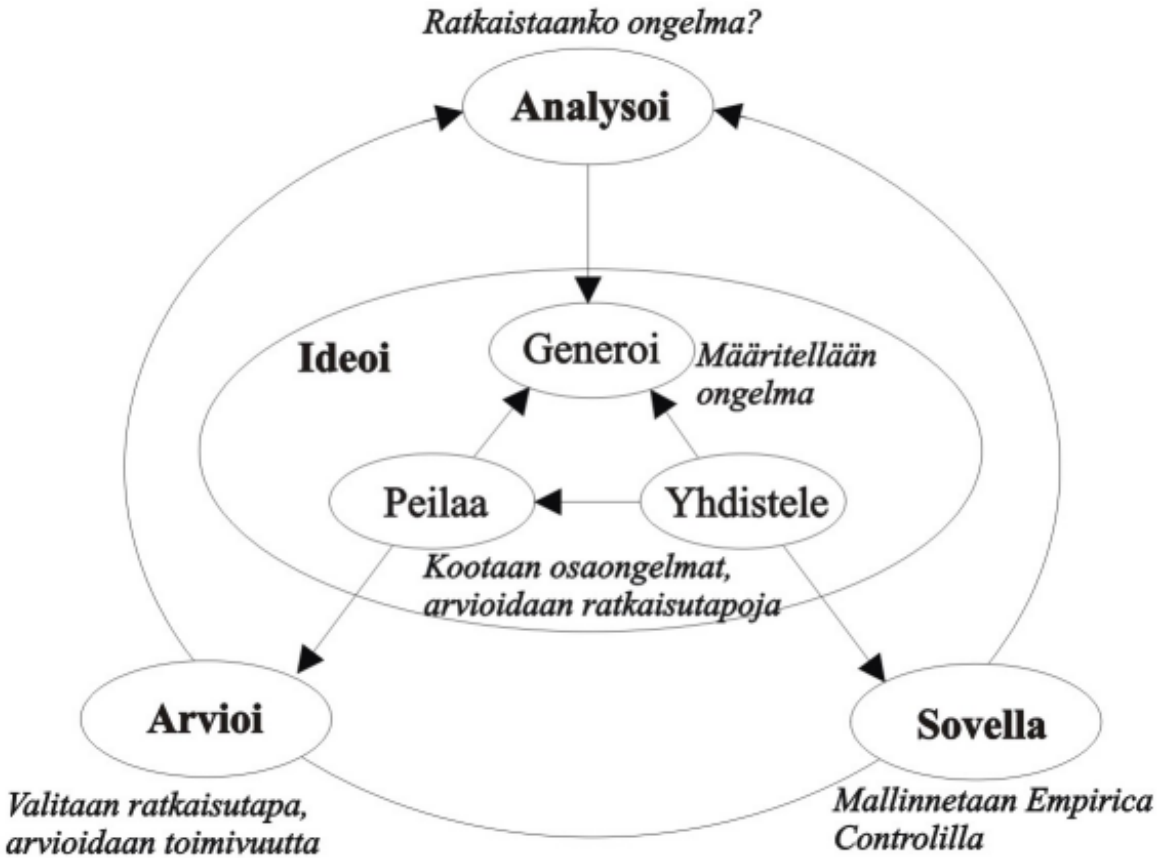
Ben-Arin (2001) mukaan tietojenkäsittelytieteen opetuksen kehittämisen ongelmana on se, ettei konstruktivismia koskevaa *oppimisparadigmaa* ole juurikaan tutkittu, toisin kuin luonnontieteiden opetuksessa. Konstruktivismin mukaan opettaja ei voi ohittaa oppilaan aiempia tietoja, vaan hänen tulee kyseenalaistaa ne tavoitteenaan ohjata oppilasta korjaamaan teorioitaan. Hän toteaa konstruktivismin yhteensopivuuden tietojenkäsittelytieteen opetukseen olevan toistaiseksi epäselvää, koska pelkästään filosofisilta lähtökohdiltaan tietojenkäsittelytiede poikkeaa radikaalisti muista tieteistä, matematiikkaa lukuun ottamatta. Konstruktivismin soveltuvuus tietojenkäsittelytieteen opettamiseen edellyttää kahden lähtökohdan huomioimista. Ensiksi oppilaalla tulisi olla kattava kokonaiskuva tietokoneesta, eli oppilaalle tulisi muodostua kognitiivinen rakenne tietokoneesta aiempien tietojensa perusteella. Tämän rakenteen varaan oppilas voi rakentaa tietämyksensä opetettavasta asiasta aistinvaraisten kokemustensa perusteella, sitomalla opitun asian itselleen konkreettisiin tietoihin. Toiseksi tietokone muodostaa saavutettavissa olevan *ontologisen todellisuuden*. Tietokoneiden avulla opetettava tieto voitaisiin näin yhdistää oppilaan todellisuuteen, jolloin oppimisesta tulisi mielekkäämpää. (Ben-Ari, 2001)

2.3 Ongelmanratkaisu

Ongelmanratkaisun avulla pyritään kehittämään ajattelua. Mehtäläinen (1992) toteaa ongelmanratkaisun tavoitteeksi saada ihminen pohtimaan ja analysoimaan ympäristöään siten, että hän kykenisi löytämään ja tätä kautta ratkaisemaan ongelmia. Hänen mukaansa ongelmanratkaisua voidaankin harjoittaa puhtaasti matemaattisten tehtävien tai jonkin todellisen tai kuvitellun tilanteen simuloinnin avulla. Luvussa 3 esitelty Empirica Control -ohjelmointiympäristö voidaan siis nähdä hyödyllisenä apuvälineenä ongelmanratkaisutaitojen harjaannuttamisessa. Lavonen et al. (2001a) toteavat suorittamassaan tapaustutkimuksessa oppilaiden tukeutuvan ongelmakeskeiseen näkökulmaan Empirica Control -ohjelmiston käytössä. He toteavat oppilaiden työskennelleen itsenäisesti Empirica Control -ohjelmiston parissa yritys-erehdys-periaatteella. Heidän mukaansa opettaja toimii tilanteessa ohjaajana.

Itsetarkkailu eli *reflektointi* ja suunnittelu ovat ongelmanratkaisun kaksi tärkeää *metakognitiivista* ominaisuutta. Itsetarkkailu viittaa yksilön kykyyn tarkistaa ongelmanratkaisuprosessin etenemistä. Tämä kyky on erityisen tärkeä kohdattaessa kunnianhimoisia tai hämmentäviä ongelmia. Suunnittelun avulla monimutkainenkin ongelma kyetään hajottamaan osaongelmiin, joiden ratkettua saavutetaan lopullinen ratkaisu (Derry & Hawkes, 1993).

Meisalon et al. (2000) mukaan luova ongelmanratkaisuprosessi voidaan ajatella syklisenä prosessina (kuva 2). Ongelmanratkaisuprosessin aikana voidaan ongelmaa tarkentaa useita kertoja. Luovaa ongelmanratkaisua suositaan nykyään kaikkialla elinkeino- ja työelämässä. Luovan ongelmanratkaisun edut on huomattu myös kouluissa, mutta menetelmän käyttöönotto on edennyt hitaasti. Luovaa ongelmanratkaisua sovelletaan myös ongelmalähtöisessä oppimisessa, jossa tavoitteena on, että oppilaat tutustuvat itsenäisesti opittavaan aiheeseen ratkoen eteen tulevia ongelmia. (Meisalo et al., 2000)



Kuva 2: Luovan ongelmanratkaisun prosessi (kts. Meisalo et al., 2000).

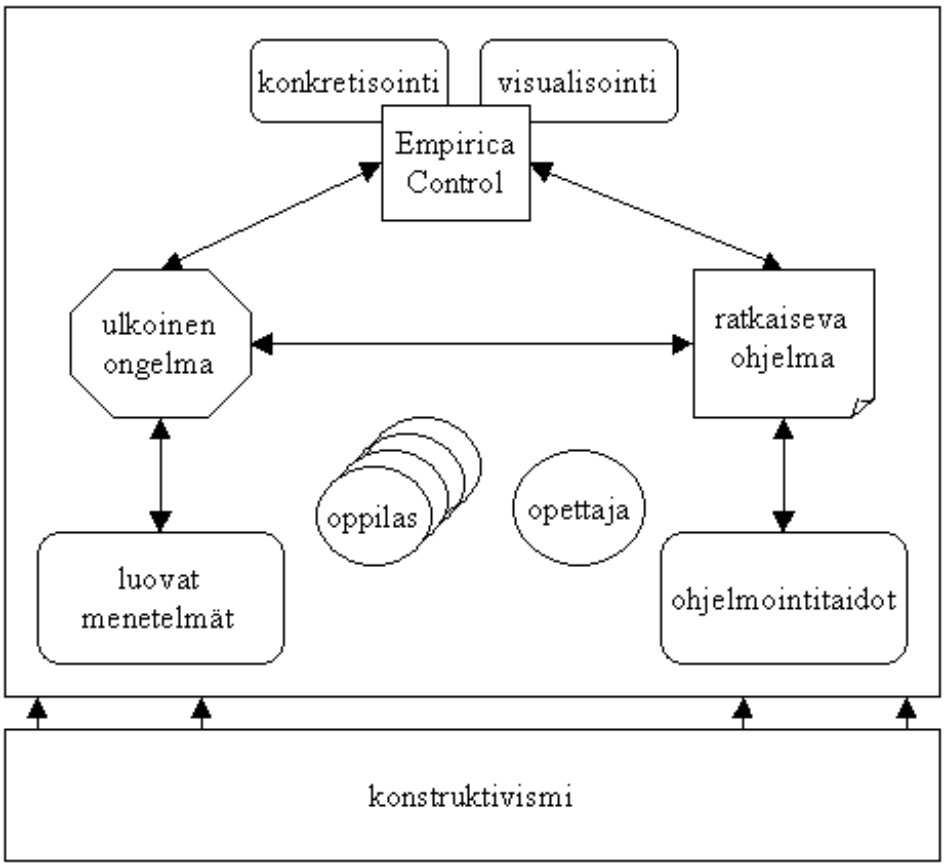
Lavonen et al. (2001a) jakavat luovan ongelmanratkaisuprosessin peräkkäisiin vaiheisiin. Aluksi määritellään ongelma, jonka jälkeen tunnistetaan ongelmaan liittyvät tosiasiat ja ase-

tetaan päämäärät. Seuraavaksi ideoidaan vaihtoehtoja ja arvioidaan syntyneitä ideoita. Viimeiseksi valitaan ratkaisu ja sitä testataan ja arvioidaan.

2.4 Ohjelmoinnin opettaminen ja Empirica Control

Perusopetuksessa käytettävälle ohjelmointikielellä määritellään neljä vaatimusta, joita ovat yksinkertaisuus, ilmaisuvoima, yhteensopivuus sekä kognitiivinen rikkaus. Ohjelmointikielen yksinkertaisuus tekee kielestä helposti ymmärrettävän, opittavan ja käytettävän. Kielen ilmaisuvoima puolestaan kertoo sen, kuinka helposti ja luonnollisella tavalla kyseistä ohjelmointikieltä voidaan käyttää olemassa olevien ongelmien ratkaisemiseen. Yhteensopivuuden avulla ohjelmointikieli voidaan sitoa useiden sovellusalueiden käyttöön ja opettamiseen. Täten uusia asioita esitettäessä jo omaksutulla ja opitulla peruskielellä voidaan oppilaiden toiminta keskittää täysin opeteltaviin asioihin eikä aikaa tarvitse käyttää opetuksessa käytettävän ohjelmointikielen rakenteiden ja käsitteiden opettamiseen. Kognitiivisen rikkauden avulla voidaan puolestaan määrittää kuinka hyvin ohjelmointikieli tukee oppilaiden omaa ajatusmaailmaa ja heidän tapaansa ratkaista kohdattuja ongelmia. (Hassinen, 1986)

Empirica Control -ohjelmistoa voidaan hyödyntää ohjelmoinnin opettamisessa monin keinoin. *Visuaalisen ohjelmoinnin* avulla voidaan vähentää oppilaan tarvetta opetella ulkoa sekä muistaa ohjelmoinnin rakenteita. Empirica Control voidaan yhdistää oppimistilanteeseen konkreettista maailmaa hyödyntävänä osana. Empirica Controlin avulla oppilas voi yhdistää opittavia asioita ympäröivään maailmaan sekä manipuloida tutkittavaa tilannetta. Oppimistilanteessa konstruktivismi tukee oppilaiden luovia ongelmanratkaisumenetelmiä sekä ohjelmointitaitojen kehittymistä. Näkemyksemme mukaan konstruktivismi tukee koko oppimisprosessia. Samalla oppimistilanteessa yhdistyvät ulkoisen maailman ongelma sekä Empirica Controlin avulla luotava ratkaiseva ohjelma. Kuvassa 3 on laatimamme kaavio Empirica Controlin hyödyntämisestä avoimessa oppimistilanteessa.



Kuva 3: Oppimistilanteessa esiintyvät tekijät.

3 VISUAALISUUS OHJELMOINNIN OPPIMISEN TUKENA *(Harri Kähkönen)*

Levyn et al. (2000) mukaan ohjelman toiminnan visualisoinnilla ja animoinnilla voidaan edistää huomattavasti oppilaan kykyä muodostaa toteuttamiskelpoinen mentaalimalli ohjelman toiminnasta. He toteavat algoritmien animoinnin olevan käyttökelpoisempaa avoimessa tehtävässä kuin suljetussa koetilanteessa. Avoimessa tehtävässä oppilaille tarjoutuu mahdollisuus tukea animointia kommunikoimalla vertaistensa kanssa. Opetuskokeilua suorittaessamme käytimme hyväksemme juuri avointa ongelman määrittelyä. Oppilaat muodostivat itse ongelmatilanteen, jonka he ratkaisivat käytössä olevien välineiden ja Empirica Control -ohjelmiston avulla. Tässä luvussa käymme läpi ohjelmoinnin oppimista edistäviä visualisoivia ja konkretisoivia menetelmiä, kuten ohjelman toiminnan visualisointia sekä niiden sitomista konkreettiseen maailmaan. Lisäksi tutustumme visualisointijärjestelmien suunnittelun perusteisiin.

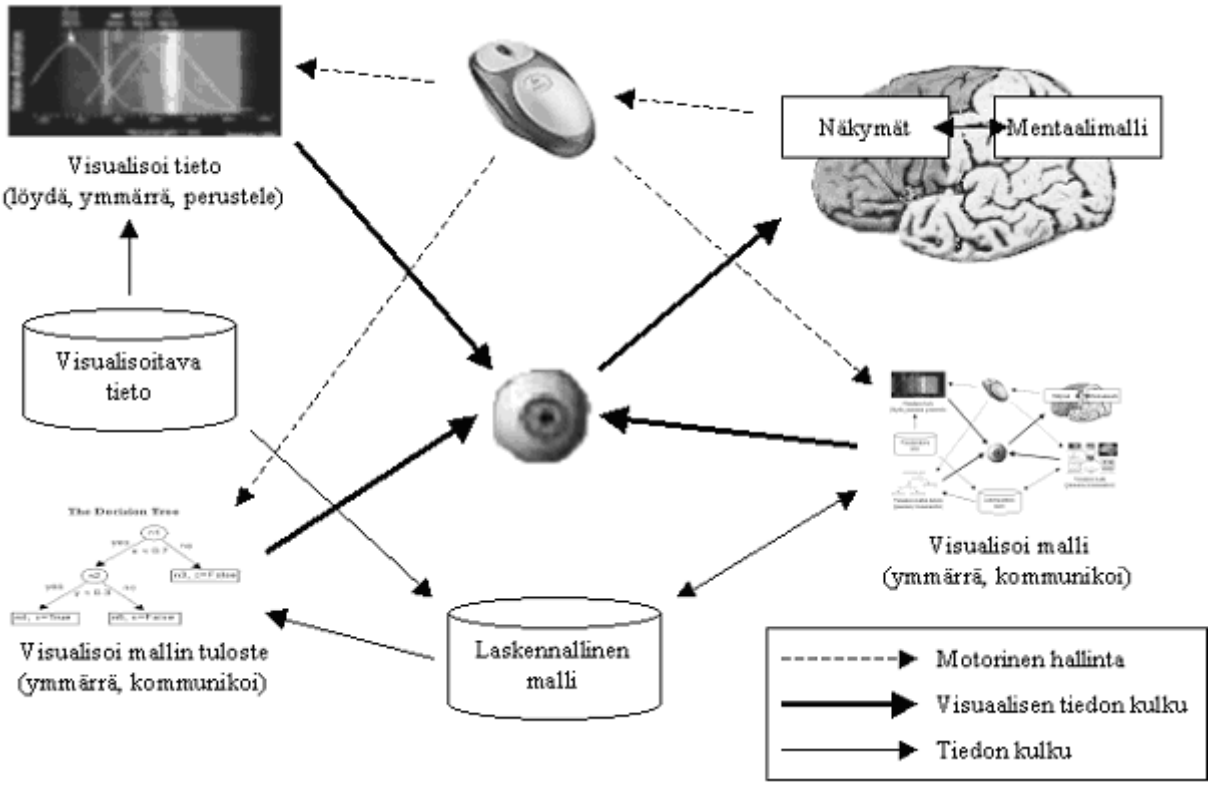
3.1 Visualisointi

Visualisointi on Cardin et al. (1999) mukaan tietokoneavusteista, interaktiivista tiedon visuaalista esittämistä *kognition* vahvistamiseksi. Heidän mukaansa kognitio on tiedon hankintaa tai soveltamista ja visualisoinnin päämääränä on oivallusten aikaansaaminen, ei niinkään kuvien tarjoaminen. Visualisointia käytetään hyväksi mallintamisessa, joka on osa ongelmanratkaisua (Waisel et al., 1999; Crapo et al., 2000). Waisel et al. näkevät visualisoinnin edistävän mentaalimallien luomista vähentämällä työmuistin tarvetta. Oikeellisen mentaalimallin muodostaminen vaatii heidän mukaansa mallien manipulointia, mikä on monimutkainen ja työmuistia vaativa tehtävä. Kuitenkin visualisointia hyödynnettäessä noviiseille tulisi tarjota mahdollisuus yksinkertaisempien visualisointien käyttöön, etteivät suuret tietomäärät hämmentäisi heitä. Jehng, Tung ja Chang (1999) sekä Perrig ja Kintsch (1985) esittävät Saariluoman (2001) mukaan visualisoinnin kielenä, jonka avulla voidaan paitsi aktivoida ideoita ja ehdottaa uusia, myös opettaa ihmisiä ajattelemaan. Tämä tapahtuu tarjoamalla tehokkaita visuaalisia käsitteitä tai niiden syvempää ymmärrystä.

Crapon et al. (2000) mukaan mallintaminen on läheinen osa ongelmanratkaisua ja se voidaan ajatella olevan osaongelmien ratkaisuun pyrkivä menetelmä. Mallintaminen voidaan luokittaa ulottuvuuksien, kuten näkyvyys, toiminta ja yleisyys mukaan. Mallintaminen, kuten ongelmanratkaisukin ovat vaativia prosesseja, joissa eksperttien ja noviisien erot tulevat esille. Vi-

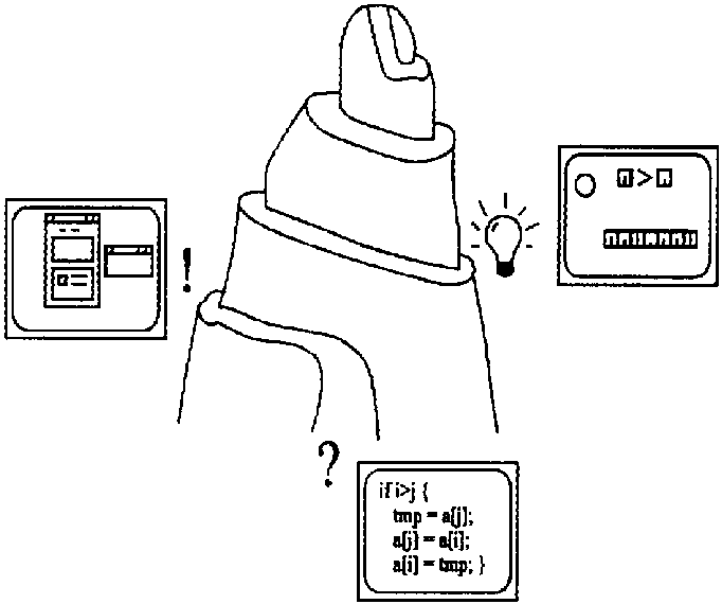
sualisointi voi osaltaan helpottaa noviisin kykyä mallintaa ongelmaa, jakaa se osaongelmiin, saavuttaa ratkaisu ja sitä kautta oikeellinen mentaalimalli.

Tietokoneavusteinen visualisointi voidaan nähdä tehokkaana mallintamisen tukena (kuva 4). Tiedon visualisointiin erikoistuneet työkalut auttavat mallintajaa tiedon sisäisten suhteiden löytämisessä ja ymmärtämisessä. Tietokoneen toiminnan visualisointi sallii käyttäjän vertailla omia mentaalimallejaan tietokoneen luomiin malleihin sekä keskustella mallin kanssa sitä muokkaamalla niin kauan kunnes tyydyttävä mentaalimalli on saavutettu. (Crapo et al., 2000)



Kuva 4: Tietokoneavusteinen visualisointi mallintamisessa (kts. Crapo et al., 2000).

Meisalo et al. (1997) kuvaavat *algoritmien animoinnin* Eliot-ohjelmiston avulla helpottavan oppimista oppimisspiraalin (kuva 5) kuvaamalla tavalla. Ensimmäisessä vaiheessa käyttäjä kirjoittaa algoritmin ohjelmakoodin, jonka jälkeen, hän toisessa vaiheessa määrittelee mitä muuttujia animoidaan. Kolmannessa vaiheessa Eliot muodostaa animaation automaattisesti. Animaation avulla käyttäjä muodostaa käsitteellisen mallin oppimastaan. Meisalon et al. kuvaamaa oppimisspiraalia voidaan siten verrata Crapon et al. (2000) määritelmään siitä, kuinka oppimista voidaan edistää visualisoinnin avulla.



Kuva 5: Eliot-oppimisspiraali (Meisalo et al., 1997).

Käsittelemme seuraavassa visualisoinnin suhdetta ohjelmointiin kahdesta varsin erilaisesta näkökulmasta: *ohjelman visualisoinnista* sekä visuaalisesta ohjelmoinnista. Nämä kaksi visualisoinnin lähestymistapaa ovat usein vaarassa sekoittua, vaikka ne eroavat toisistaan selkeästi.

3.1.1 Ohjelman visualisointi

Ohjelman visualisointia kuvataan monin eri tavoin. Myersin (1986) mukaan ohjelman visualisointi on ohjelman tai sen jonkin osan esittämistä graafisesti sen suorituksen yhteydessä. Romanin ja Coxin (1992) mukaan ohjelman visualisointi esittää ohjelman toiminnan graafisesti. Petren et al. (1998) määritelmän mukaan ohjelmien visualisoinnin tarkoituksena on löytää yksinkertaisuus monimutkaisesta kokonaisuudesta. Ohjelmien visualisointi on kuitenkin toisenlaista toimintaa, kuin muu ohjelmien seuraamismenetelmä. He toteavatkin visualisointijärjestelmän esittävän ohjelman graafisena esityksenä. Voimmekin nähdä ohjelmien visualisoinnin helposti irrallaan esimerkiksi tekstinkäsittelyohjelman tai pelin ulkoasusta. Ohjelmien visualisointi liittyy kiinteästi ohjelmien toteutukseen ja toimintaan.

Myers (1986) kuvaa ohjelmien visualisoinnille erilaisia tasoja. Visualisointi voi olla staattista ohjelmakoodin visualisointia, joka esittää ohjelmakoodin lähinnä vuokaavion muodossa käyttäen hyväkseen tekstin muotoiluja, kuten lihavointia ja kursivointia. Ohjelmakoodin visualisointi voi olla dynaamista, jolloin visualisointi keskittyy lähinnä osoittamaan mitä kohtaa

koodista ollaan suorittamassa. Empirica Control -ohjelmistolla laaditut ohjelmat täyttävät ohjelman staattisen ja dynaamisen visualisoinnin ominaisuudet. Ohjelmat muodostuvat visuaalisiksi vuokaavioiksi, joiden seuraamista suorituksen mukaisesti etenevä sininen pallo helpottaa. Myersin mukaan tiedon staattisen visualisoinnin tavoitteena on helpottaa ohjelman virheenkorjausta. Ohjelman tietorakenteet kuvataan graafisessa muodossa suhteineen, jolloin ohjelman sisäiset suhteet ovat helpommin ymmärrettävissä. Dynaaminen tiedon visualisointi luo ohjelman suorituksesta animaatioita, jotka helpottavat ohjelman suorituksen ymmärtämistä ja seuraamista. Empirica Control -ohjelmistossa tiedon visualisointia ei juurikaan tapahdu, koska monimutkaisia tietorakenteita ei voida laatia. Ohjelmissa käytettävät syötetiedot saadaan Empirica Interfacen kautta antureilta, joiden mittaamat arvot saadaan näkyviin erillisiin ikkunoihin. Mittausarvojen muutokset voivat vaikuttaa sinisen pallon etenemiseen, jolloin käyttäjän on helppoa seurata syötetietojen muutosten vaikutusta ohjelman suoritukseen.

Petre et al. (1998) näkevät visualisoinnilla olevan etuja niin ohjelmoinnin ammattilaisille kuin aloitteleville ohjelmoijille. Ammattilaiset tarvitsevat visualisointia ulkoistaakseen mielikuviansa, käyttääkseen visualisointia keskustellakseen vertaistensa kanssa esiintyvistä ongelmista tai eräänlaisena ajatustyökaluna. Ajatustyökaluna visualisointi toimii tiedon täydentäjänä ja sen korostajana tarjoten eräänlaista ulkoista muistia, toisenlaisia näkökulmia ongelmiin, ongelman karttaa sekä hahmotusapua erilaisten alaongelmien suhteista kokonaisuuteen. Aloitteleville ohjelmoijille tällaiset visualisoinnit voisivat tarjota silmäyksen eksperttien ajatusmaailmaan ja ajattelun prosesseihin. Samalla aloittelijat näkisivät, kuinka ohjelmat toimivat sekä kokisivat erilaisen tavan ongelman kuvaamiseen. Petre et al. toteavat kuitenkin, ettei visualisoinnilla saavuteta hyötyä, ellei sen käyttötarkoitusta, merkitystä sekä rajoituksia tiedosteta. Normaali luokkaopetus vaatii samanlaisen huomion. Hyvää oppimistulosta ei saavuteta ilman oppitunnin huolellista valmistelua ja toteutusta. Visualisoinnin muodostamista ja tehokasta käyttöä käsitellään tarkemmin luvussa 3.2.

Aloittelevan ohjelmoijan ongelma on muodostaa käyttökelpoinen mentaalimalli tietokoneen toiminnasta. Tämä ongelma voi vaikeuttaa ohjelmoinnin perustehtäviä, kuten ohjelman etenemisen seuraamista (Ben-Ari, 2000). Levy et al. (2000) näkevätkin ohjelmien ja algoritmien animoinnin ja visualisoinnin mahdollistavan tällaisten mentaalimallien huomattavan kehittymisen. He kuitenkin toteavat, etteivät empiiriset tutkimukset yksiselitteisesti tue tätä näke-

mystä. Heidän mukaansa ohjelmien animoinnista ei kuitenkaan ole hyötyä kaikille. Heikompia ohjelmoijia uusi työkalu voi hämmentää ja kokeneemmat ohjelmoijat eivät sitä tarvitse.

Empirica Control -ohjelmisto toteuttaa ohjelman visualisoinnin periaatteita. Ohjelman suorituksen aikana sininen pallo liikkuu ruudulla visuaalisesti kuvatun ohjelman suorituksen mukaisesti. Samaan aikaan Empirica-mittausjärjestelmään liitettyjen antureiden arvot saadaan näkyville erillisiin ikkunoihin, joiden avulla ohjelman suorituksen etenemistä voidaan analysoida.

3.1.2 Visuaalinen ohjelmointi

Romanin ja Coxin (1992) mukaan visuaalinen ohjelmointi on ohjelman laatimista graafisin määritelmin. Myersin (1986) määritelmä on samankaltainen. Hänen mukaansa visuaalinen ohjelmointi viittaa mihin tahansa ohjelmistoon, jolla käyttäjä voi määritellä ohjelman kaksi- tai useampiulotteisesti. Hänen mukaansa visuaalinen ohjelmointi sisältää vuokaaviot ja graafiset ohjelmointikielet, joiden avulla ohjelman toiminnan etenemistä kuvataan. Empirica Control -ohjelmisto antaa käyttäjälleen mahdollisuuden kaksiulotteiseen visuaaliseen ohjelmointiin Myersin määritelmät toteuttaen. Empirica Control -ohjelmiston käyttöliittymä mahdollistaa ohjelman rakentamisen graafisten kuvakkeiden avulla. Kuvakkeet viittaavat ohjelmointirakenteisiin, kuten silmukkaan tai ehtolauseeseen. Ohjelma rakentuu kuvakkeista vuokaavioon, josta ohjelman toimintaa ja etenemistä voidaan seurata.

Visuaalinen ohjelmointi sisältää kuitenkin ongelmia, joita ei ole helppoa ratkaista. Käsiteltäessä suuria määriä tietoa visuaalinen ohjelmointi vaatii huomattavasti enemmän tilaa kuin perinteinen ohjelmakoodi (Myers, 1986). Tämä on nähtävissä myös Empirica Control -ohjelmistossa. Helposti laadittavat lauseet ja ehdot vievät huomattavasti enemmän tilaa kuin perinteisellä ohjelmakoodilla ilmaistuna. Visuaaliset ohjelmat voivat lisäksi olla tehottomia ja toiminnallisuudeltaan rajoittuneita (Myers, 1986). Empirica Control -ohjelmiston käyttöä rajoittaa lähinnä Empirica Interfaceen liitettävien antureiden määrä. Antureita voi olla yhtäaikaisesti vain kaksi.

Myers (1986) näkee yhtenä visuaalisen ohjelmoinnin ongelmana ohjelman kommentoinnin hankaluuden. Empirica Control -ohjelmistossa kommentointi on mahdollistettu lisäämällä selventäviä tekstikenttiä ohjelmakaavioon. Kommentit eivät kuitenkaan liiku ohjelman muuttu-

essa vaan jäävät alkuperäiseen paikkaansa. Tällainen ohjelman kommentointi ainoastaan sekoittaisi ohjelman ymmärtämistä. Myers näkee vielä hankalien ohjelmien staattisen esittämisen visuaalisen ohjelmoinnin vaikeutena. Empirica Control -ohjelmistolla ei saavuteta niin hankalia ohjelmia, etteikö graafinen ohjelmaa kuvaava kaavio sen selventäisi. Ongelmaksi voi muodostua vain ohjelman näkyvyys. Pienellä näytöllä ohjelmasta saadaan näkyviin liian pieni osa. Empirica Control -ohjelmisto sisältää mahdollisuuden pienentää ohjelmakaaviota, jolloin näkyviin saadaan suurempi osa ohjelmaa. Kaavion pienentäminen kuitenkin heikentää ohjelman luettavuutta, koska ohjelmakoodin kuvakkeiden tarkkuus heikentyy pienennettäessä.

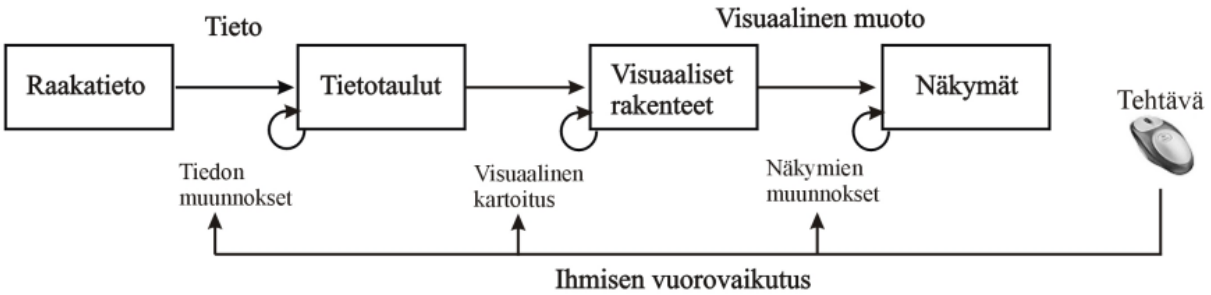
Taulukossa 2 kuvaamme, kuinka visuaalisen ohjelmoinnin ongelmat esiintyvät mielestämme Empirica Control -ohjelmistossa. Taulukossa vasen sarake sisältää Myersin (1986) määrittelemät visuaalisessa ohjelmoinnissa esiintyvät ongelmat sekä oikeassa sarakkeessa kuvaamme esiintymisen Empirica Controlissa. Kuvauksemme pohjautuvat opetuskokeilujen aikana saatuihin kokemuksiin.

Visuaalisen ohjelman ongelma	Esiintyminen Empirica Controlissa
Vaatii enemmän näyttötilaa	Empirica Controlin kuvakkeet vievät huomattavasti tekstiä enemmän tilaa näytöllä, jolloin ohjelman näkyvyys voi heikentyä.
Tehottomuus ja toiminnallinen rajoittuneisuus	Empirica Control suunniteltu luonnontieteiden opetukseen, muunlaisten ongelmien ratkaisu vaikeaa.
Ohjelman kommentointi hankalaa	Empirica Controlissa asetetut kommentit eivät ole sidottuja ohjelman rakenteeseen. Ohjelman muuttuessa kommentit eivät liiku rakenteiden mukana.
Hankalien ohjelmien staattinen esittäminen	Empirica Controlissa ohjelmista ei muodostu niin monimutkaisia ettei niiden staattinen esittäminen onnistuisi. Ongelmaksi voi muodostua vain näkyvyys.

Taulukko 2: Havaitsemamme visuaalisen ohjelmoinnin ongelmat (Myers, 1986) Empirica Controlissa.

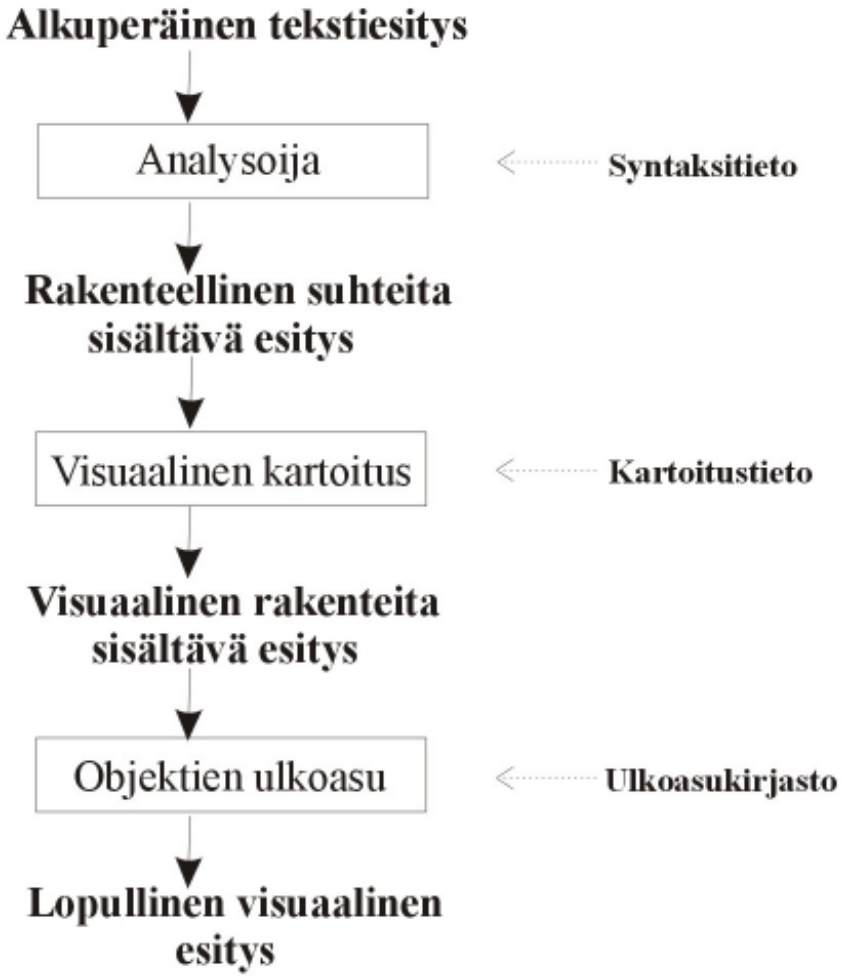
3.1.3 Visualisoinnin muodostaminen

Tiedon visualisointi voidaan muodostaa kartoittamalla tieto erillisten vaiheiden kautta visuaaliseen muotoon (kuva 6). Ensimmäisen vaiheen raakatieto (ohjelmakoodi) muunnetaan tietotauluiksi. Käyttäjä voi vaikuttaa muunnokseen tehtävänsä edellyttämällä tavalla. Tietotauluista muodostetaan käyttäjän määritysten avulla visuaalinen kartoitus. Lopuksi käyttäjä määrittelee visualisoitavalle tiedolle näkymät, jotka tietokone muodostaa. Visualisoinnin muodostamista ohjaavat käyttäjän tarpeet eli suoritettavana oleva tehtävä. (Card et al., 1999)



Kuva 6: Tiedon visualisoinnin malli (kts. Card et al., 1999).

Kamada ja Kawai (1991) ovat määritelleet yleisen kehyksen visualisointien muodostamiselle (kuva 7). Heidän mukaansa visualisointi muodostetaan neljän vaiheen avulla. Ensimmäisellä tasolla on perinteinen tekstimuodossa oleva tieto, esimerkiksi perinteinen ohjelmakoodi. Seuraavassa vaiheessa tekstimuodossa oleva tieto muunnetaan suhteita sisältävään rakenteelliseen esitykseen. Kolmannessa vaiheessa rakenteellinen esitys suhteineen kuvataan visuaalisten objektien avulla. Lopuksi muodostetaan valmis visuaalinen esitys tiedosta. Kamadan ja Kawain määritelmä visualisoinnin muodostamiselle käsittää samankaltaiset vaiheet kuin Cardin et al. (1999) malli tiedon visualisoinnista (kuva 6). Empirica Control -ohjelmisto muodostaa visualisoinnin automaattisesti, jolloin käyttäjän toimet eivät siihen vaikuta. Visualisoinnin muodostuminen kuitenkin riippuu käyttäjän muodostaman ohjelman rakenteesta sekä erillisissä ikkunoissa esitettävistä ominaisuuksista (lokikirjan esitys, antureiden mittaustulokset).



Kuva 7: Yleinen kehys visualisoinnin muodostamiselle (kts. Kamada & Kawai, 1991).

3.2 Visualisoinnin tehokas käyttö

Visualisointijärjestelmille kuten muillekin informaatiojärjestelmille asetetaan vaatimuksia, joiden täyttäminen on ehtona niiden tehokkaalle käytölle. Seuraavassa käsittelemme visualisointijärjestelmien käytettävyysvaatimuksia, kognitiivisten ulottuvuuksien vaikutusta käytettävyyteen Empirica Control -ohjelmistossa sekä visualisoinnin opetuskäyttöä.

3.2.1 Visualisointijärjestelmän käytettävyys

Visualisoinnin suunnittelu keskittyy usein grafiikan, värien, käyttöliittymän sekä elementtien näytölle asettelun huomioimiseen. Liian usein ohitetaan visualisoinnin tehokkaan opetuskäytön vaatimat seikat, kuten käyttäjien tarpeet, tehtävät, tarvittava tieto sekä suunnittelijoiden saatavilla olevat resurssit. Käyttäjien kattava analysointi tulisi olla suunnitteluprosessin alustava askel. Kaikille käyttäjille sopivaa visualisointia ei ole olemassa. Esimerkiksi kokenut ohjelmoija vaatii järjestelmältä erilaista tiedon syvyyttä ja saatavuutta sekä laajempia mahdollisuuksia vaikuttaa järjestelmän toimintaan. Visualisointijärjestelmää ei voida hyödyntää tehokkaasti eri käyttötarkoituksiin, koska luennolla ja kokeessa käytettävän järjestelmän vaatimukset eroavat toisistaan. (Khuri, 2001)

Huonolla visualisointijärjestelmän suunnittelulla aiheutetaan monenlaisia tarkkaavaisuuteen liittyviä virheitä. Visualisoinnissa käytetyt elementit voivat esimerkiksi kadota samankaltaiseen taustaan (Saarenheimo, 2001). Käyttäjän kadottaessa osan visualisoinnin tarjoamasta tiedosta, hän voi muodostaa virheellisiä malleja, eikä tavoiteltua oppimistulosta saavuteta. Visualisointijärjestelmän sisältämä tieto tulee esittää tavalla, jonka käyttäjät kykenevät ymmärtämään. Khurin (2001) mukaan järjestelmän käyttäjien aika ei saa kulua kuvien tarkoituksen pohtimiseen vaan monimutkainen tieto tulisi hajottaa pienemmiksi, helpommin ymmärrettäviksi osakokonaisuuksiksi. Tiedon pilkkominen sisältää omat vaaransa, mutta monimutkaisten ja suurten tietomäärien kohdalla se on välttämätöntä. Saarenheimo toteaakin, että tehokkaan visualisoinnin luomisen suurin ongelma on tärkeimmän tiedon esittäminen helposti käsiteltävässä muodossa.

Visualisointijärjestelmän suunnittelussa tulee huomioida ihmisten havainnoinnin rajoitteet. Tärkeät tiedot tulisi sijoittaa näytön vasempaan laitaan yläosaan ja erityyppinen tieto tulisi sijoittaa omiin alueisiinsa. Järjestelmän suunnittelussa tulee huomioida lisäksi visualisoinnin

vaatima tila. Kaikki ylimääräiset komponentit vähentävät visualisoinnille varattua tilaa ja vaikeuttavat visualisoinnin ymmärtämistä (Khuri, 2001). Brownin (1988) mielestä Khurin mukaan visualisoinnin tukena tulisi esittää myös vaihtoehtoinen tieto, kuten ohjelmakoodi jota visualisoidaan. Khuri muistuttaa, että visualisointijärjestelmää laadittaessa tulisi huomioida myös ihmisen kyky havaita eri värejä sekä niiden yhdisteitä. Vahvat kontrastit kuten punaisen ja vihreän yhdistelmä muodostavat olemattomia varjoja ja jälkikuvia. Värien käytössä on ymmärrettävä myös niiden perinteiset tarkoitukset. Punainen kuvaa vaaraa, keltainen varovaisuutta ja vihreä hyväksyntää. Värejä tulisi käyttää myös ryhmittämään samankaltaista tietoa, kuten ohjelmakoodin suorituksessa muuttujaa tai ehtoa.

Tiedon tehokas prosessointi vaatii muistia. Ihmisen työmuisti rajoittuu tallentamaan neljästä seitsemään tietoelementtiä kerrallaan. Todennäköisemmin tämä määrä on neljä kuin seitsemän. Usein järjestelmien suunnittelijat eivät huomioi tätä rajoitetta, jolloin järjestelmän käytettävyys kärsii. Muistikuormaa lisää myös järjestelmän monimutkainen navigointi tai tiedon etsiminen kuvasta. Ihmisen muistin ylimääräinen kuormittaminen vähentää tiedon prosessointiin tarvittavaa kapasiteettia ja osaltaan vaikeuttaa oppimista. (Saarenheimo, 2001)

3.2.2 Kognitiiviset ulottuvuudet Empirica Control -ohjelmistossa

Greenin ja Blackwellin (1998) määrittelemien kognitiivisten ulottuvuuksien tarkoituksena on toimia keskusteluvälineinä, joiden avulla käyttäjät voivat arvioida järjestelmän käytettävyyttä. Kognitiiviset ulottuvuudet keskittyvät käytettävyyden ominaisuuksiin, jotka vaikeuttavat järjestelmän oppimista ja käytettävyyttä. He määrittelevät käyttäjien toiminnan järjestelmissä neljään kategoriaan: tiedon lisäämiseen, syöttämiseen, muokkaamiseen ja kokeilevaan suunnitteluun. Tietynlainen toiminta järjestelmässä vaikuttaa sen kognitiivisiin ulottuvuuksiin. Taulukossa 3 on kuusi kognitiivista ulottuvuutta joiden avulla Empirica Control -ohjelmistoa voidaan mielestämme arvioida.

Kognitiivinen ulottuvuus	Selite
Viskositeetti (viscosity)	Valmiin työn muuttamisen vaikeus. Kun tahdot muuttaa aiempaa työtä, kuinka vaikeaa se on?
Piilotetut riippuvuudet (hidden dependencies)	Muutosten teko jossain osassa vaikuttaa toiseen osaan, mutta yhteys ei ole näkyvissä.
Ennakkoon sitoutuminen (premature commitment)	Pakottaako järjestelmän notaatio toimimaan tietyllä tavalla, vai voiko käyttäjä toimia haluamallaan tavalla?
Abstraktio (abstraction)	Onko käyttäjän mahdollista määritellä uusia toimintatapoja järjestelmän notaation sisällä?
Toissijainen notaatio (secondary notation)	Onko mahdollista ilmaista tietoa, joka ei ole järjestelmän notaation mukaista?
Näkyvyys (visibility)	Kuinka helppoa on nähdä järjestelmän eri osia niitä luotaessa tai muokattaessa?

Taulukko 3: Kuusi kognitiivista ulottuvuutta (kts. Green & Blackwell, 1998).

Empirica Control -ohjelmistossa viskositeetti tulee esiin tilanteessa, jossa halutaan asettaa suurempi ohjelmarakenne silmukkaan. Empirica Control -ohjelmistossa silmukka muodostuu valmiiksi ja sen sisälle tulevat rakenteet pitää lisätä aina jälkikäteen. Tällainen *notaatio* pakottaa käyttäjän myös sitoutumaan ennakkoon tiettyyn ohjelmointitapaan. Greenin ja Blackwellin (1998) mukaan järjestelmän viskositeettia voidaan vähentää lisäämällä abstraktiota. Empirica Control -ohjelmistossa abstraktiota voitaisiin lisätä esimerkiksi sallimalla aliohjelmien laatiminen. Kuitenkin abstraktion lisääminen edellyttää sitä, että käyttäjä tutustuu abstraktioiden toimintaan ennakkoon. Abstraktion lisääminen voi myös aiheuttaa piilotettuja riippuvuuksia abstraktioiden välillä. Empirica Control -ohjelmistossa ongelmaksi muodostuu myös ohjelman näkyvyys. Kuvakkeiden muodostuminen vuokaavioon vie huomattavasti enemmän tilaa kuin perinteinen ohjelmakoodi. Myös näkyvyyttä voidaan parantaa lisäämällä abstraktiota.

3.2.3 Visualisointi opetuksessa

Visualisointia on käytetty ja tutkittu aktiivisesti ohjelmoinnin opetuksessa vasta parin vuosikymmenen ajan (Miraftabi, 2001). Lisäksi visualisointia on hyödynnetty ohjelmien selittämisessä, suunnittelussa ja analysoinnissa 1980-luvulta lähtien (Khuri, 2001). Visu-

alisoinnin hyödyntämistä opetuksessa hidastaa kuitenkin materiaalien laatimisessa vaadittavat resurssit. Miraftabi näkee visualisoinnin käytön mahdollisina ongelmina niiden vaatimat tekniset resurssit sekä opettajien haluttomuuden laatia visualisointeja oppimateriaaleistaan. Khurin mukaan visualisointi edistää oppimista, sillä se ylläpitää oppilaiden keskittymiskykyä, selventää monimutkaisia käsitteitä, automatisoi esimerkkejä ja esityksiä sekä kannustaa käytännölliseen oppimisprosessiin. Hänen mukaansa visualisointi lisää myös kommunikaatiota opettajien ja oppilaiden välillä sekä vapauttaa oppimaan tekemällä ilman pelkoa virheistä.

Petren et al. (1998) mukaan visualisoinnin hyödyntäminen algoritmien opetuksessa vaatii muutakin kuin ohjelmarivien esittämistä. He esittävät kaksi olettamusta tiedon visualisoinnin tehokkaasta opetuskäytöstä. Ensimmäiseksi tiedon etsiminen ja käyttäminen on helpompaa, kun niiden välille muodostetaan kognitiivinen yhtenäisyys. Toiseksi visualisoinnin avulla voidaan tuoda esiin tiedon olennaiset piirteet. Tällöin kognitiivista yhtenäisyyttä tiedon epäolennaisten ominaisuuksien kanssa ei saavuteta. Näiden kahden olettamuksen toteuttamiseksi tulisi selvittää millaisia ohjelmointitehtäviä visualisoinnin avulla halutaan opettaa, kuinka näiden tehtävien oppimista voitaisiin tukea ja kuinka tehtävät muuttuvat visualisointijärjestelmien muuttuessa.

3.3 Ohjelmoinnin konkretisointi

Papertin (1980) mukaan opetuksessa voidaan saavuttaa huomattavasti parempia tuloksia, jos tieteelliselle ajattelulle ja oppijan tärkeiksi kokemille toiminnoille löydetään yhteys. Empirica Control toteuttaa Papertin käsitystä oppimisen tehostamisesta ohjelmoinnin opettamisessa samoin kuin LOGO-ympäristö matematiikan opettamisessa. Seuraavassa käsittelemme *ohjelmien konkretisointia* sekä *konkretisoivaa ohjelmointia*.

3.3.1 Ohjelman konkretisointi

Ohjelman konkretisointia voidaan edesauttaa kehittämällä fyysisiä olioita, jotka heijastavat luodun ohjelman toimintaa. Hyvä esimerkki ohjelmien konkretisoinnista on LEGO/LOGO-ohjelmointiympäristö (Järvinen, 1998), jossa legoilla rakennettuja robotteja ohjataan ohjelmien avulla – ohjelmien toiminnasta saadaan näin konkreettista palautetta.

Empirica Controlilla rakennettu ohjelma muodostuu valmiiksi vuokaavioksi. Vuokaaviossa mahdolliset ohjelman haarautumiset, esimerkiksi ehtolauseessa, kuvataan sivuaskelina ohjel-

man suorituksessa. Empirica Controlilla laadittujen ohjelmien suoritusta voidaan helposti seurata sinisen pallon avulla. Empirica Controlin avulla ohjelmoidessa ohjelmointi konkretisoituu myös Empirica Interfacen kautta saatavien mittaustulosten manipulointina. Käyttäjä voi tutkia tietyn mittausarvon muutoksen vaikutusta ohjelman toimintaan, sinisen pallon liikkumiseen vuokaaviolla. Sininen pallo yhdistää myös ehtolauseiden toteutumisen fyysiseen maailmaan muodostamalla reitilleen liikennevaloja sen mukaan, toteutuuko tietty ehto vai ei.

3.3.2 Konkretisoiva ohjelmointi

Empirica Controlin avulla ohjelmointi voidaan nähdä konkretisoivana – ohjelmaa laadittaessa voidaan ajatella, että ohjelma on sinisen pallon ohjausta. Kun käyttäjä ajattelee, että sininen pallo on ohjattava näyttelijä hän oppii ohjelmoinnin perusteet nopeammin (Brusikovsky et al., 1994 Lavosen et al., 2001 mukaan). Konkretisoiva ohjelmointi voidaan nähdä ohjelman rakentamisena fyysisistä objekteista. Ohjelmointi voisi tapahtua esimerkiksi tuttujen rakennuspalikoiden avulla, joita yhdistelemällä muodostuisi toimiva ohjelma.

Empirica Control -ohjelmiston avulla ohjelmoinnin oppiminen voidaan sitoa fyysiseen maailmaan. Oppilaat voivat ratkaista konkreetteja ongelmia Empirica Controlilla ohjelmoiden ja manipuloiden mahdollisia ohjelman käyttämiä mittausarvoja. Empirica Control -ohjelmistolla toimiessaan he voivat rakentaa oheistarvikkeiden, kuten rakennuspalikoiden avulla oman maailmansa, jonka ongelmia he voivat ratkaista. Oppilaiden ongelman muodostaminen fyysisten elementeistä sekä sen ratkaiseminen Empirica Controlin avulla toteuttaa Papertin (1980) käsitystä oppimisen tehostamisesta. Oppilaat kokevat itse luomansa maailman tärkeäksi, jolloin ongelmien ratkaiseminen motivoi heitä enemmän kuin perinteiset koulutehtävät. Tällaisessa oppimistilanteessa lapset toimivat hänen ajatustensa mukaisesti tieteen tekijöinä, eivät tiedon passiivisina vastaanottajina.

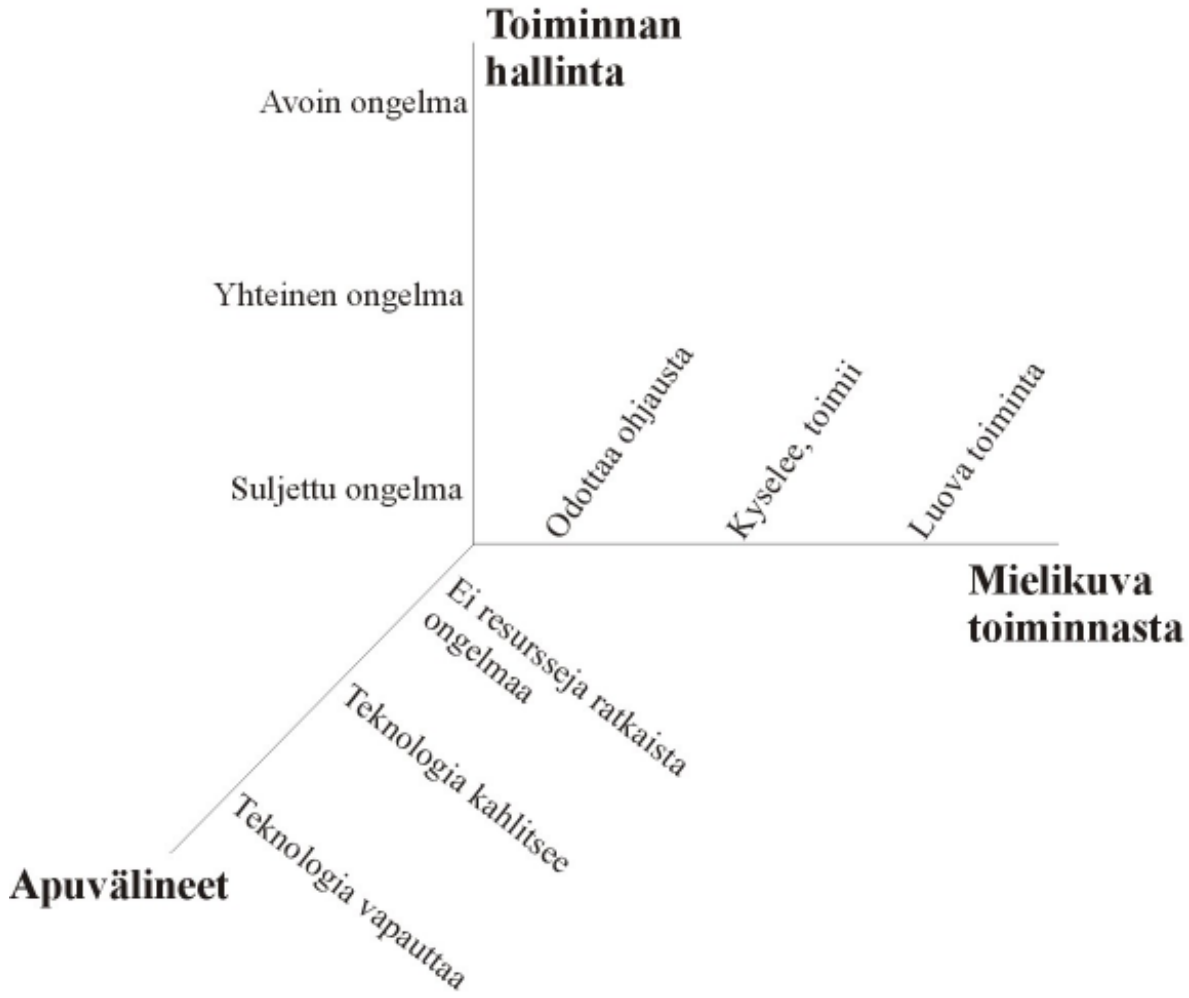
4 EMPIRICA OPETUKSESSA *(Kai Piiroinen)*

Tässä luvussa tutustumme Empirica Control -ohjelmiston käyttöön ohjelmoinnin opetuksessa. Tämän lisäksi tutustumme Empirica Controlin erilaisiin käyttömahdollisuuksiin opetuksessa.

4.1 Avoin oppimisympäristö

Oppimisympäristöllä kuvataan kokonaisuutta, jossa oppiminen tapahtuu. Se sisältää opettajan, oppilaan, opetusmateriaalit ja -välineet. Oppimisympäristö on avoin, kun opiskelijalla on mahdollisuus valikoida ne välineet ja materiaalit, joiden avulla hän kykenee sisäistämään opiskeltavan kokonaisuuden (Meisalo et al., 2000). Avoin oppimisympäristö siis jakaa vastuuta oppimisesta opettajan ja oppilaan välillä. Opittavaa asiaa ei näin ollen mekaanisesti syötetä oppilaiden opittavaksi, vaan oppilailla on mahdollisuus vaikuttaa oppimisprosessin etenemiseen. Meisalon et al. mukaan oppimisympäristössä opettajan ja oppilaan roolit muuttuvat. Opettajasta tulee pikemminkin ohjaaja ja oppilas seikkailee oppimisympäristössä aktiivisesti tietoa keräten.

Oppimisympäristön avoimuudelle on määritelty eri ulottuvuuksia. Lavosen ja Meisalon (1994) mielestä Meisalon et al. (2000) mukaan avoimuutta voidaan analysoida kolmen ulottuvuuden suhteen. Esitämme ulottuvuudet Meisaloa et al. mukaillen (kuva 8). Ensimmäinen ulottuvuus, *toiminnan hallinta*, korostaa tehtävien muotoilun, suunnittelun ja suorittamisen vapautta oppilaan näkökulmasta. Toisena ulottuvuutena määritellään *mielikuva toiminnasta*, jonka mukaan oppimisympäristön avoimuus vaatii oppilaalta tiettyjä pohjatietoja ja -taitoja, jotta oppilas osaisi suorittaa oppimisympäristön tehtävät. Kolmantena ulottuvuutena on *apuvälineet*, joka kohdistaa kiinnostuksemme siihen, vapauttavatko vai kahlitsevatko käytettävissä olevat välineet ja niiden tarjoamat vaihtoehdot oppimista. Toisin sanoen, vapauttaako vai vangitseeko teknologia oppilaan ajattelua.



Kuva 8: Oppimisympäristön ulottuvuudet (kts. Meisalo et al., 2000).

4.2 Empirica

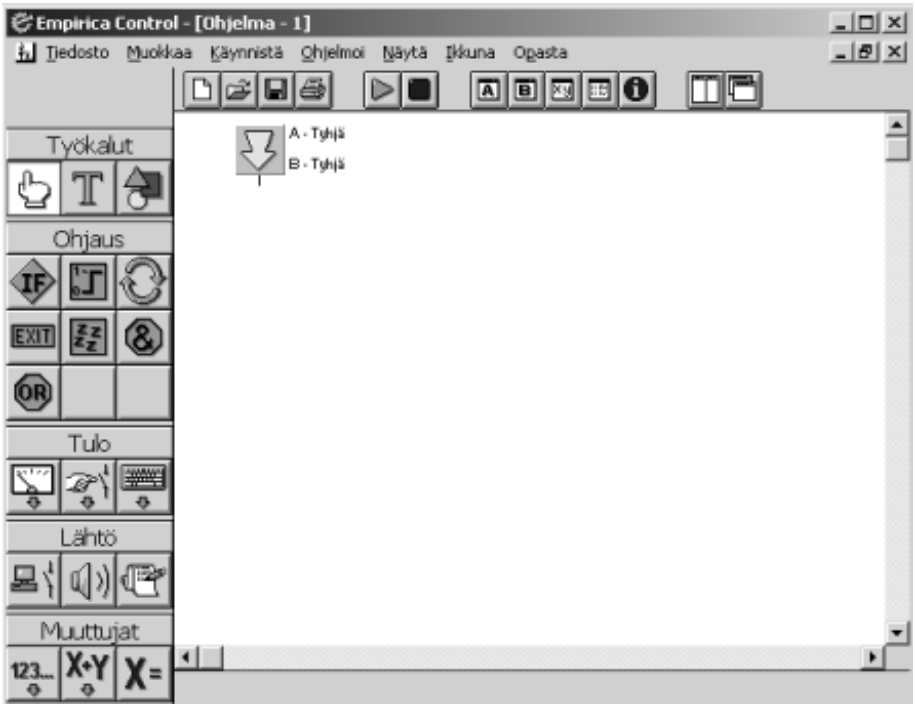
Empirica-järjestelmän kehittäminen tapahtui vuorovaikutuksessa LUONTI-projektin (Luonnontieteiden Opetuksen Teknologiset Innovaatiot) kehittämishankkeen kanssa. LUONTI-projekti oli Helsingin yliopiston opettajankoulutuslaitoksen ja Taloudellisen Tiedotustoimiston vuonna 1995 käynnistämä yhteinen luonnontieteiden ja teknologian opetuksen tutkimus-, kehittämis- ja koulutushanke. LUONTI-projekti oli monitieteellinen ja se pyrki tutkimaan ja kehittämään oppimisympäristöjä, joissa teknologia tukee oppilaiden oppimista ja tekee siitä samalla kiinnostavampaa ja monipuolisempaa. (Lavonen et al., 1996)

Empirica on siis mittausjärjestelmä luonnontieteiden ja teknologian opetukseen. Empirica toimii Windows-ympäristössä ja näin ollen sen käyttöliittymä on kehitetty mahdollisimman joustavaksi, helppokäyttöiseksi ja yhdenmukaiseksi tavallisimmin käytettyjen työvälineoh-

jelmien kuten Microsoft Excelin ja Wordin kanssa, joita opettajat ja oppilaat käyttävät koulumaailmassa. Empiricaa voidaan näin ollen käyttää työvälineenä, jonka avulla oppilaat voivat kerätä konkreettisesti luonnosta monipuolista ja luotettavaa tietoa sekä samalla esittää keräämänsä tiedon sellaisessa muodossa, että sitä on helppoa tulkita ja analysoida. Kerätyt mittaustulokset voidaan esittää näytöllä halutussa muodossa, joko digitaalisesti ja / tai graafisesti. (Lavonen et al., 1996)

4.3 Empirica Control osana oppimisympäristöä

Osana Empirica-järjestelmää toimii visuaalinen ohjelmointiympäristö Empirica Control (kuva 9), jonka avulla voidaan mallintaa konkreetit tai abstraktit ongelmat ja tutkia luotua mallia. Seuraavassa käsitellään mallien hyödyntämistä opetuksessa sekä Empirica Controlia osana oppimistilannetta.



Kuva 9: Empirica Controlin käyttöliittymä.

4.3.1 Mallit opetuksessa

Malleja hyödynnetään opetuksessa kuvaamaan konkreettista tai abstraktia kohdetta. Malleista ei koskaan saada identtisiä kohteensa kanssa. Niiden avulla voidaan siitä huolimatta helpottaa ja edistää oppilaiden käsitystä opittavasta aiheesta. Meisalo et al. (2000) määrittelevät ope-

tuskäyttöön tarkoitetuille malleille tiettyjä vaatimuksia, jotka keräsimme seuraavaan taulukkoon (taulukko 4).

Mallin ominaisuus	**Selite**
Kattava	Mallin tulisi kuvata kohteen nykytilaa, mutta sen tulisi myös antaa ennusteita kohteesta.
Fokusoiva	Mallin tulisi kuvata kohteen keskeisiä piirteitä epäolennaisten kustannuksella.
Läpinäkyvä	Mallin tulisi paljastaa kohteen sisäinen looginen rakenne ja systeemin sisäiset vuorovaikutukset.
Konkreetti	Mallin tulisi tarjota visuaalinen mielikuva kohteestaan.
Pohjatiedolle rakentuva	Mallin tulisi käyttää hyväkseen aiemmin opittuja malleja.
Oikeellinen	Mallille tulisi määrittää selkeä pätevyysalue.
Ymmärrettävä	Mallin luomisessa tulisi käyttää esitystapaa ja käsitteitä, jotka kohderyhmä (l. oppilaat) ymmärtävät.

Taulukko 4: Opetuskäyttöön tarkoitettujen mallien vaatimuksia (kts. Meisalo et al., 2000).

Empirica Control -ohjelmiston avulla voidaan opetuskäyttöön laatia malleja konkreettisista reaalimaailman tilanteista. Oppilaille tarjotaan lisäksi mahdollisuus tutkia sitä, kuinka muutokset systeemissä vaikuttavat muihin mallin oleellisiin piirteisiin. Empirica Controlin avulla voidaan esimerkiksi tutkia, kuinka oven avaaminen vaikuttaa huoneen lämpötilaan. Mallien avulla oppilaat pääsevät käsiksi konkreettisiin tilanteisiin, joiden avulla uusi tieto voidaan yhdistää aikaisempaan (kts. luku 2).

4.3.2 Empirica Control oppimistilanteessa

Empirica Control tarjoaa oppimistilanteessa mahdollisuuden yhdistää reaalimaailman konkreetteja ilmiöitä oppimistilanteeseen. Empirica Control -ohjelmiston käyttäminen osana oppimisympäristöä tarjoaa mahdollisuuden konkreettien ongelmien analysointiin ja ratkaisemiseen. Meisalo et al. (2000) näkevät tämän ominaisuuden olennaiseksi määrittäessään oppimisympäristön suunnittelun avointa torimallia. Torimallissa oppija vaeltaa tiedon torilla pysähtyen kiinnostavilla kojuilla. Kojuista oppija valikoi itseään kiinnostavat tiedot. Tiettyä

metaforaa noudattava oppimisympäristö määrittelee käyttäjänsä roolin ja toimintaympäristön. Oppimisympäristön kyky tuoda reaalimaailman ilmiöt käsiteltäväksi tekee siitä avoimen.

4.4 Empirica Control -ohjelmiston käyttö opetuksessa

Koulumaailmassa on usein korostettu yhtä oikeaa ratkaisua ongelmaan. Tämä voi pahimmassa tapauksessa lannistaa oppilasta, vaikka oppilas pitäisi saada ponnistelemaan ratkaisun löytämiseksi. Papertin (1980) mukaan lapsen intuition vääräksi todistaminen ei palvele oppimista. Hän toteaakin, että LOGO-ympäristössä toimiessaan myös opettaja oppii oppilaiden kanssa. Kun oppilaat tiedostavat tämän, he alkavat kokea opettajan enemmän vertaisenaan kuin auktoriteettina.

Ohjelmoinnin opetus peruskouluissa rajoittuu pitkälti opettajien omaan aktiivisuuteen ja oppilaiden kiinnostukseen. Empirica Control -ohjelmiston graafinen käyttöliittymä tarjoaa helpomman pääsyn ohjelmoinnin ja tietotekniikan maailmaan (vertaa käyttöjärjestelmä) vähentäen myös opettajilta vaadittua ohjelmointitaitoa. Empirica Control -ohjelmistolla ohjelmointi tapahtuu graafisen käyttöliittymän avulla ja konstruktivistinen oppiminen konkretisoituu, kun oppilaat manipuloivat fyysistä ympäristöään mallintamaansa järjestelmää varten.

Anderson ja Naps (2001) määrittelevät erilaisia algoritmien ymmärtämisen tasoja (taulukko 5). Kun ymmärtämisen tasot on määritelty tarkasti, voidaan visualisointien suunnittelussa hyödyntää tehokkaasti pedagogisia näkökulmia. He määrittelevät myös tasot visualisointien suunnittelulle opetuskäyttöön (taulukko 6).

Taso	Algoritmin ymmärtämisen aste
Taso 1	**Ymmärretään algoritmi reseptinä:** Oppilaat ymmärtävät, että algoritmi etenee askeleittain ja miksi algoritmi ratkaisee tietyn ongelman. He osaavat seurata askeleita tyypillisellä aineistolla sekä kykenevät ennakoimaan, miten algoritmi reagoisi erikoiseen aineistoon.
Taso 2	**Ymmärretään kuinka algoritmi voidaan sijoittaa ohjelmaan:** Oppilaat osaavat kirjoittaa ohjelman, joka hyödyntää algoritmia.
Taso 3	**Algoritmin tehokkuuden analysointi:** Ymmärretään, kuinka aineiston koko vaikuttaa suoritus- ja tilavaativuuksiin.
Taso 4	**Algoritmin oikeellisuuden todistaminen:** Oppilaat osaavat matemaattisin keinoin todistaa algoritmin oikeellisuuden.

Taulukko 5: Algoritmien ymmärtämisen tasot (kts. Anderson & Naps, 2001).

Taso	Opetuksellisen suunnittelun aste
Taso 1	**Algoritmin visualisointi ilman vuorovaikutusta:** Animaatio kuin televisio-ohjelma – pedagogisesti vain huomionherättäjä. Animaatio voi olla sulavaa tai sarja peräkkäisiä kuvia.
Taso 2	**Selittävät tekstimateriaalit animoinnin tukena:** Vaatimuksena minkä tahansa ymmärryksellisen tason saavuttaminen. Materiaali voi olla staattista (aina sama riippumatta visualisoinnin vaiheesta), riippuvaista algoritmista (materiaali vaihtuu algoritmin suorituskohdan mukaan) tai dynaamista (materiaali huomioi visualisoinnin tilan sekä muuteltavat arvot).
Taso 3	**Mahdollista suunnitella omaa aineistoa algoritmilla:** Mahdollista tunnistaa algoritmin käyttäytyminen parhaassa ja huonoimmassa tapauksessa.
Taso 4	**Oppilaalle esitetään kysymyksiä ja pyydetään ennustamaan ohjelman toimintaa:** Tekstimateriaalit voivat helpottaa oppilasta muodostamaan toivotun yhteyden algoritmin suorittamisen ja visuaalisten kokonaisuuksien välille. Halutun tuloksen voi saavuttaa myös pysäyttämällä visualisointi ja pakottamalla oppilas ennustamaan sen toimintaa.
Taso 5	**Nähtävillä olevan informaatiomäärän säätely:** Oppilaan kyettävä valitsemaan oppimisen kannalta sopivimmat näkymät, jolloin liian suuret tietomäärät eivät hämmennä oppilasta.
Taso 6	**Mahdollisuus palata takaisinpäin visualisoinnissa:** Jos jotain kokonaisuutta ei ymmärretä, voidaan palata takaisin visualisoinnissa ja katsoa se uudelleen.
Taso 7	**Oppilas suunnittelee visualisoinnin itse:** Tason kaksi ymmärrys saavutetaan parhaiten kun oppilas suunnittelee oman visualisointinsa.
Taso 8	**Läheinen suhde visualisointijärjestelmän ja kurssimateriaalin välillä:** Visualisointijärjestelmän yhdistäminen kurssiin voi vaikuttaa enemmän vuorovaikutuksen syntymiseen kuin systeemin ominaisuudet.

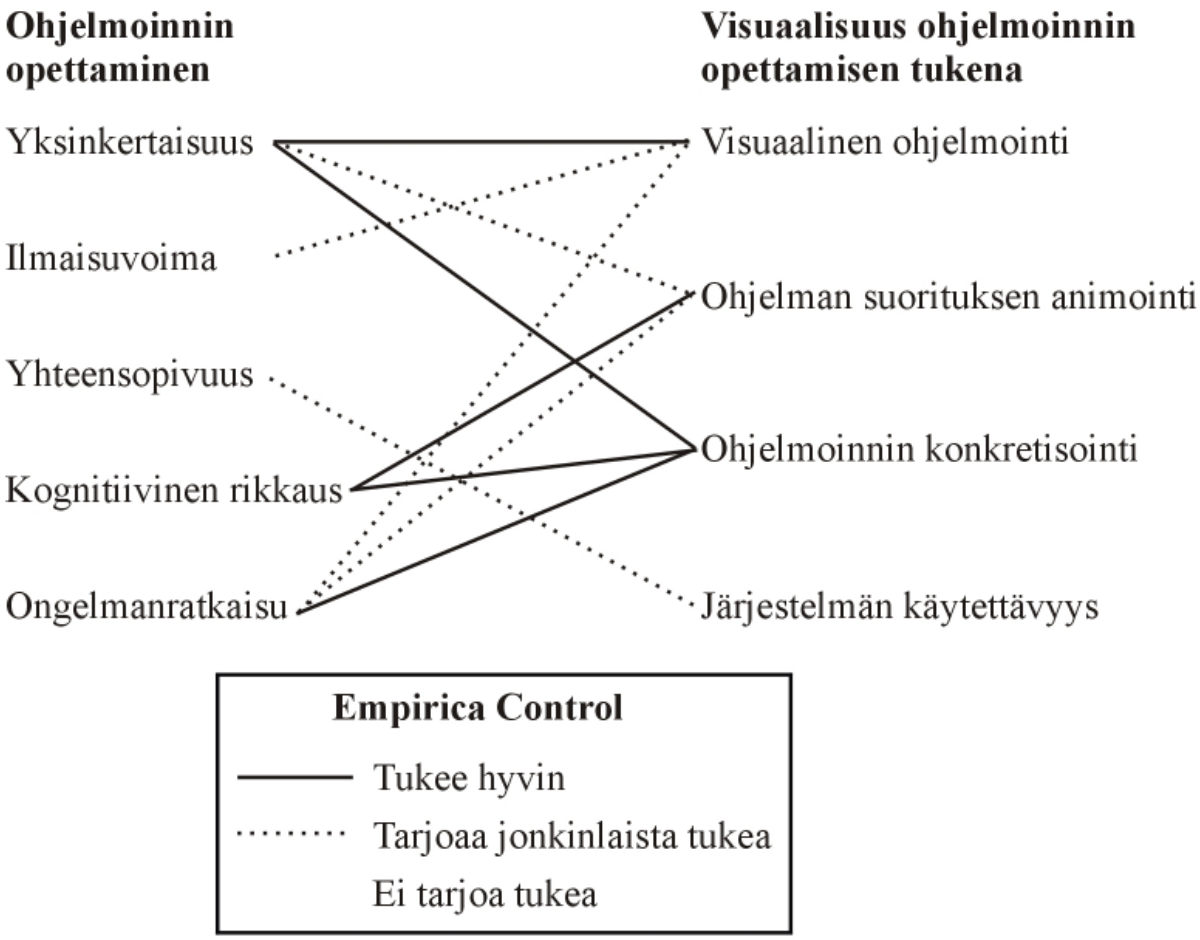
Taulukko 6: Opetuksellisen suunnittelun tasot (kts. Anderson & Naps, 2001).

4.5 Empirica Control ja ohjelmointi

Empirica Control -ohjelmiston kehitystyön tarkoituksena oli kehittää työkalu, joka minimoisi opettajan suoran ohjauksen sekä ohjelmointisääntöjen muistamisen tarpeen. Tällä tavalla oppilaan resurssit vapautuisivat konstruktivistiseen ja luovaan toimintaan (Lavonen et al., 2001a). Empirica Control -ohjelmisto vähentää ulkoa muistamisen tarvetta juuri ikonipohjaisuutensa takia. Kaikki ohjelmointirakenteita kuvaavat ikonit ovat selkeitä ja kuvaavia, jolloin oppilas voi päätellä mikä ikoni tarkoittaa mitäkin. Tämä on olennainen parannus, sillä aiemmissa tutkimuksissa on todettu, että esimerkiksi LOGO-ympäristössä oppimista vaikeuttaa ohjelmointikielen syntaksi (Järvinen, 1998).

Lavonen et al. (2001a) toteavat, että visuaalinen ohjelmointityökalu helpottaa oppilaiden työskentelyä. Heidän mukaansa Empirica Control -ohjelmiston ikonit ja parametrien asettelut ikkunoiden kautta vähentävät oppilaiden muistitaakkaa vapauttaen resursseja itse työskentelyyn. Empirica Control -ohjelmistossa ohjelman muodostaminen tapahtuu lisäämällä ikoneita ohjelmapohjalle. Nämä ikonit muodostavat helposti seurattavan vuokaavion. Ohjelmien suorituksen seuraamista puolestaan helpottaa sininen pallo, joka liikkuu vuokaaviota pitkin kertoen samalla ohjelman suorituksen etenemisen. Lavonen et al. toteavatkin oppilaiden kykenevän ajattelemaan ohjelmointia juuri tämän sinisen pallon ohjailuna. Tämä helpottaa myös ohjelman virheenkorjausta sekä arviointia. Oppilas voi jatkuvasti seurata sitä, mitkä ohjelmarakenteet tulevat suoritetuiksi ja kuinka ne vaikuttavat ohjelman toimintaan.

Empirica Control sisältää piirteitä, jotka yhdistävät ohjelmoinnin oppimista (luku 2) sekä sitä tukevaa visualisoinnin käyttöä (luku 3). Empirica Controlin käytöstä ohjelmoinnin opettamisessa on mielestämme suurta apua juuri sen tarjoamien visuaalisten ominaisuuksien vuoksi. Empirica Controlin avulla voidaan korostaa yhteyksiä, jotka ovat tärkeitä ohjelmoinnin oppimisen kannalta. Kuvaamme ohjelmoinnin opettamisessa tärkeiden piirteiden yhteyttä visuaalisuuteen Empirica Controlia käyttäen kuvassa 10.



Kuva 10: Empirica Controlin ohjelmoinnin oppimista tukevia tekijöitä.

4.5.1 Ohjelmointiympäristön ominaisuudet

Empirica Control -ohjelmisto tarjoaa ohjelmoijalle visuaalisen ympäristön, jossa ohjelmointi tapahtuu valitsemalla kuvakkeita loogiseen järjestykseen, jolloin ohjelmapohjalle rakentuu vuokaavio. Samalla ohjelman generaattori tuottaa itse ohjelman. Visuaalinen ohjelman kuvaus helpottaa ohjelman rakenteen ymmärtämistä ja ohjelman etenemisen seuraamista. Empirica Control antaa käyttöön myös lokikirjan, muuttujaluettelon sekä digitaalisen mittarin, joiden avulla oppilas kykenee seuraamaan eri arvojen muutoksia eri tilanteissa. Empirica Controlin ohjelmointiympäristö tukee kokeellista tiedonhankintaa, toiminnallisuutta ja yhteistoiminnallisuutta. Nämä piirteet ovat ominaisia esimerkiksi luonnontieteiden ja teknologian opetuksessa.

Empirica-mittausjärjestelmä varustettuna Empirica Control -ohjelmistolla sisältää piirteitä, joiden avulla voidaan tukea ohjelmoinnin opettamista. Kuvaamme taulukossa 7, kuinka nämä piirteet tukevat erilaisten ohjelmoinnin perusrakenteiden oppimista ja opettamista.

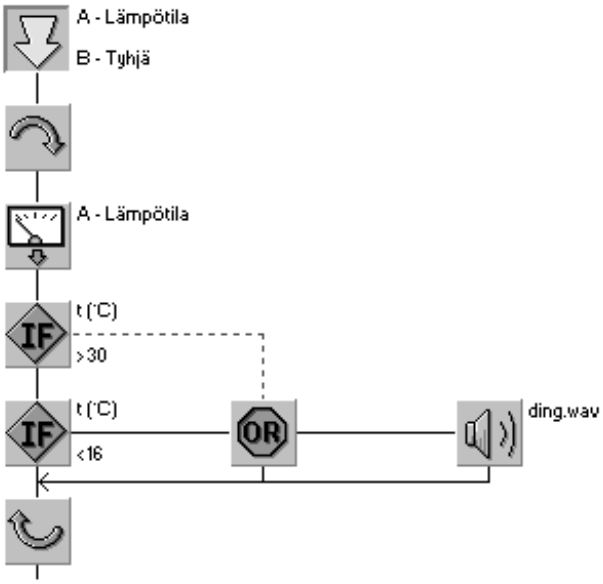
Ohjelmoinnin osa-alue / Empirica -mittausj. piirre	Ehtolause	Muuttuja	Silmukka (toistolause)	Loogiset operaatiot
Sininen pallo	Muodostaa ikonin viereen "liikennevalot" sen mukaan, toteutuuko ehto vai ei.	Ei merkittävää apua.	Sininen pallo kulkee silmukassa niin kauan kuin on määritelty. Jos silmukka suoritetaan n-kertaa, pallon liikkuessa suoritettujen kertojen lukumäärä ilmaistaan ikonin vieressä.	Muodostaa "liikennevalot" erikseen molemmille ehdoille sekä loogiselle operaatiolle riippuen siitä toteutuvatko ne.
Konkretisointi antureiden avulla	Voidaan ratkaista todellisuuteen liittyviä ongelmia, kuten lämpötilan säätely.	Esimerkiksi ohjelman lähtöarvoa (kuten lämpötila) voidaan muuttaa antureiden avulla.	Antureiden avulla voidaan seurata ohjelman etenemistä silmukan eri rakenteissa. Lisäksi voidaan tutkia mahdollisen lopetusehdon toteutumista.	Kuten ehtolauseella.
Visuaalinen ohjelmointi	Ehtolauseen voi helposti hiiren avulla lisätä ohjelmarunkoon valitsemalla rakennetta kuvaavan ikonin.	Muuttujan lisäys ohjelmaan kuten ehtolauseen. Muuttujien määrittely hankalampaa (ohjelman aloitusnuolen kautta).	Mahdollisuus vaikuttaa toistojen määrään helposti ominaisuuksia muuttamalla.	Luodaan lisäämällä kaksi ehtolausetta, jonka jälkeen ne yhdistetään valitulla loogisella operaatiolla.
Ohjelman vuokaavio	Ehtolauseen lisääminen aiheuttaa ohjelman vuokaavion jakautumisen kahteen vaihtoehtoiseen polkuun.	Lisättäessä muuttuja ohjelmaan asettuu se vuokaavioon osaksi ohjelmaa - ei aiheuta haarautumisia.	Silmukan rakenne muodostuu ohjelmaan lisättäessä valmiiksi. Silmukan sisällevoidaan muodostaa haluttuja rakenteita.	Looginen operaatio muodostuu ohjelmaan selkeästi yhdistäen kaksi ehtolausetta. Ehtojen väliin voi lisätä haluttuja rakenteita.
Käyttöliittymän muistettavuus	Ehtolausetta kuvaa **IF**-kuvake (JOS), joka helpottaa muistettavuutta ohjelmoidessa. Rakenne muodostuu valmiiksi, eikä sitä tarvitse opetella ulkoa.	Muuttujaa kuvaava ikoni on matematiikasta tuttu "**X=**". Tuttu rakenne helpottaa muistettavuutta.	Silmukkaa kuvaa toista mista kuvaava nuoliympyrä. Silmukasta poistutaan **exit**-kuvakkeella tai sen suoritusten lukumäärä on ennalta asetettu. Ei tarvitse muistaa suoritusehtoja.	Loogiset rakenteet **AND** (JA) sekä **OR** (TAI) ovat helposti muistettavia. Ehtojen liittäminen voi tuottaa hankaluuksia.

Taulukko 7: Ohjelmoinnin oppimista tukevat piirteet Empirica-mittausjärjestelmässä.

4.5.2 Ohjelman luominen

Empirica Control -ohjelmiston avulla ohjelma voidaan rakentaa visuaaliseksi kokonaisuudeksi. Ohjelman luominen tapahtuu lisäämällä Empirica Control -ohjelmiston käyttöliittymän (kuva 8) vasemmasta reunasta ohjelmarakenteiden kuvakkeita valmiille ohjelmapohjalle. Ohjelma rakentuu valmiiksi vuokaavioksi, jossa ohjelman suoritus etenee ylhäältä alas. Lisättä-

essä ohjelmaan ehtolauseita tai loogisia operaatioita vuokaavioon syntyy sivuaskeleita, jotka kuvaavat ohjelman suorituksen haarautumista (kuva 11).



Kuva 11: Empirica Control -ohjelmistolla luotu esimerkkiohjelma.

Empirica Control -ohjelmiston avulla ohjelmoinnin oppimisesta muodostuu helpompaa kuin perinteisellä ohjelmointikielellä. Empirica Control -ohjelmistolla ohjelmoitaessa kielen syntaksia ei tarvitse opetella ulkoa, vaan ohjelman rakenne muodostuu automaattisesti loogiseen järjestykseen. (Lavonen et al., 2001b)

Liitteessä 14 on Empirica Control 1.1a -oppimateriaali CD-levylle tallennettuna. Valmistimme oppimateriaalin harjoitustyönä. Tarkoituksenamme oli hyödyntää sitä opetuskokeiluissa. Opetuskokeilujen tiukan aikataulun vuoksi materiaalia ei kuitenkaan ehditty hyödyntää osana opetusta. Oppimateriaalin tavoitteena on perehdyttää käyttäjä Empirica-mittausjärjestelmään sekä Empirica Control -ohjelmistoon. Oppimateriaalissa selvitetään lisäksi ohjelmoinnin perusrakenteet sekä yleisiä asioita ohjelmoinnista. Materiaali sisältää myös videoleikkeitä, joissa selvitetään ohjelmointia Empirica Control -ohjelmistolla.

5 TUTKIMUSASETELMA

Tutkimuksemme lähtökohtana on selvittää, kuinka Empirica Control -ohjelmiston avulla voitaisiin opettaa ohjelmoinnin perusrakenteita. Tietotekniikan opetusta tutkivat tiedostavat ohjelmoinnin oppimisen vaikeudet (Ben-Ari, 2001). Opetuskokeilujen perusteella pyrimme selvittämään, voidaanko Empirica Controlin avulla helpottaa ohjelmoinnin oppimista ja opetusta, sekä vaikuttaa myönteisesti oppilaiden suhtautumiseen ohjelmointia kohtaan. Pyrimme tutkimuksellamme osoittamaan, että ohjelmoinnin opettaminen voitaisiin aloittaa jo perusasteella.

Perinteisen ohjelmoinnin opetuksen ensimmäinen askel on hallita ohjelmointikielen syntaksi. Empirica Controlin avulla ohjelmoiminen vapauttaa ulkoa muistamisen tarpeesta. Opetuskokeiluissa käytimme ongelmakeskeisen oppimisen ideaa. Ryhmillä oli käytössään Empirica-mittausjärjestelmä Empirica Control -ohjelmistolla varustettuna. Ryhmät määrittelivät avoimen ongelman jonka halusivat Empirica Controlin avulla ratkaista.

Tutkimuksemme tärkein ongelma on selvittää, tuoko Empirica Controlin hyödyntäminen etuja ohjelmoinnin perusrakenteiden oppimiseen. Tutkimuksessa selvitämme, kuinka oppilaat toimivat ryhmänä ratkoessaan itse määrittelemiään ongelmia Empirica Controlin avulla. Opetuskokeiluissa muodostimme ohjelmointitaidoiltaan eritasoisia ryhmiä joiden toimintaa tarkkailimme. Selvitämme myös, millaista vuorovaikutusta ryhmien sisällä tapahtuu sekä millaisia vaikutuksia aiemmalla ohjelmointikokemuksella oli ryhmissä toimimiseen.

6 TUTKIMUSMETODIT *(Harri Kähkönen)*

Tässä luvussa selvitämme tutkimuksemme keskeiset tutkimusmetodit. Käymme läpi opetuskokeilun toteutussuunnitelman, sekä tutkimukseemme vaikuttaneen *kvalitatiivisen* ja *kvantitatiivisen* osuuden.

6.1 Kvasikokeellinen tutkimus

Tutkijoiden siirtyessä ulos laboratorioista luonnollisempiin tutkimusympäristöihin, tutkimukseen vaikuttavien muuttujien kontrollointi vaikeutuu. Kvasikokeellinen tutkimus on perinteisen laboratoriokokeen tapainen, mutta olosuhteiden hallitsemattomuus ja tutkijan rooli toiminnan havainnoijana on huomioitava. Tutkijoilla on vähän, jos ollenkaan, mahdollisuuksia vaikuttaa tutkimuksen tilanteisiin. Opetuskokeilussa pyrimme tarjoamaan oppilaille mahdollisimman luonnollisen toimintaympäristön. Kuitenkin Empirica Control oli kaikille tuntematon. Käyttämämme tutkimusmetodi oli kvasikokeellinen. Kvasikokeellisessa tutkimuksessa tutkijat hyödyntävät ympäristön tarjoamia mahdollisuuksia. Tulokset pohjautuvat tutkijoiden keräämiin havaintoihin, eikä aineiston analysoinnissa voida hyödyntää laboratorioissa saatavia mittaustuloksia. Lisäksi kvasikokeellinen tutkimus on pitkäkestoisempi kuin laboratoriokokeet. (Denscombe, 1998)

Opetuskokeilu toteutettiin Empirica Control -ohjelmiston avulla. Tavoitteena oli selvittää, kuinka kyseinen ohjelmisto edesauttaa ohjelmoinnin perusrakenteiden oppimista. Tutkimuksen edetessä seurasimme ryhmien toimintaa sekä vuorovaikutussuhteita. Opetuskokeiluun osallistuville oppilaille tehtiin selväksi tutkimuksen motiivit ja aineiston keräämisen tavat. Tavoitteena oli irtaantuminen tavallisesta koulurytmistä ja oppilaita aktivoitiin tuomaan esittämään omia näkemyksiään ja ideoitaan. Tällä tavoin saavutimme lähemmän kontaktin oppilaisiin opetuskokeilun aikana. Bogdan ja Biklen (1998) toteavat, että kohderyhmän hyväksyntä on tutkimusongelmien ratkaisemisen ehto. Oppilaille selvitettiin, ettei tutkimuksen tarkoituksena ole tutkia yksilöiden vaan ryhmien toimintaa.

Laadullisessa tutkimuksessa myös tutkija osallistuu tutkittavansa maailmaan, vallitseviin tilanteisiin. Toisaalta hän pysyttelee siitä irti. Tutkija voi osallistua kohteen tai kohderyhmän toimintaan, mutta rajoitetummin ja ilman kilpailua arvostuksen tai aseman suhteen. Tutkija on tilanteessa kuin vieras, joka ei ole tullut opettamaan kohdettaan, vaan pikemminkin oppimaan

häneltä. Opetuskokeilun edetessä osallistuimme ryhmien toimintaan mahdollisimman vähän. Kun oppilaat esittivät meille kysymyksiä pyrimme aktivoimaan heitä ajattelemaan ryhmässä tarjoamatta esiintyviin ongelmiin valmiita ratkaisumalleja. Kuitenkin jos ongelmat muodostuivat ylitsepääsemättömiksi autoimme ryhmää etenemään. Tarkoituksenamme oli estää ryhmän turhautuminen ongelman edessä. (Bogdan & Biklen 1998)

Bogdan ja Biklen (1998) esittävät tutkijan osallistumisen asteelle kaksi ääripäätä. Täydellisessä seuraamisessa tutkija ei osallistu kohderyhmänsä toimintaan millään tasolla. Hän saattaa siis seurata toimintaa esimerkiksi yksisuuntaisen lasin lävitse. Toinen ratkaisu on täydellinen osallistuminen. Tällöin tutkijan ja kohderyhmän toiminnassa on havaittavissa vain pieniä eroja. Molemmat tutkimustavat ovat äärimmäisiä ja tutkijat pysyttelevät yleensä niiden välimaastossa. Opetuskokeilun aikana pyrimme välttämään ryhmien toimintaan sekaantumista. Ongelmatilanteissa pyrimme johdattelevien vastakysymysten avulla saamaan ryhmän pohtimaan mahdollisia ratkaisuvaihtoehtoja.

Opetuskokeilun edetessä mallintamisvaiheeseen ryhmät tarvitsivat enemmän tukea toiminnassaan. Ennen ongelmanmäärittelyä ja mallintamista pidetyillä oppitunneilla pyrittiin toimimaan siten, että oppilaat osallistuvat mahdollisimman paljon tuntien etenemiseen. Ohjelmoinnin perusrakenteita opetettaessa oppilaita pyydettiin esittämään todellisia tilanteita, joihin käsiteltyjä asioita voisi liittää. Mallintamisvaiheessa oppilaat tarvitsivat enemmän tukea, sillä laitteisto ja Empirica Control -ohjelmisto olivat heille tuntemattomia. Oppituntien aikana ei ehditty syventyä riittävästi Empirica-mittausjärjestelmään. Sen vuoksi oppilaat eivät kyenneet toimimaan täysin itsenäisesti. Opetuskokeilun edetessä meiltä vaadittiin vaihtelevaa osallistumista ryhmien toimintaan. Bogdan ja Biklen (1998) toteavatkin ettei valmiita, kattavia ohjeita kenttäkokeilujen suorittamiseen ole: useat näkökulmat jäävät tutkijan itsensä ratkaistavaksi.

6.2 Opetuskokeilu

Tutkimuksemme perustuu opetuskokeiluihin, jotka toteutettiin Lieksassa Lieksan lukiolla ja Merilän perusasteella. Tutkimuksessamme kiinnitimme huomiota oppilaiden toimintaan ryhmässä sekä Empirica Control -ohjelmiston käyttöön. Seurasimme ryhmien vuorovaikutustilanteita ja etsimme ryhmätyöhön aktiivisimmin ja passiivisimmin osallistuvat henkilöt. Ryhmien toimintaa seurasimme videolta, ruudunkaappauksista sekä havainnoimalla. Videointia ja

ruudunkaappauksia hyödynnettiin ryhmien ongelmanmäärittelyn ja mallintamisprosessin aikana. Näiden keinojen avulla kerätty materiaali muodostui tutkimuksen merkittävimmäksi aineistoksi kattavuutensa ansiosta. Havainnoinnin tavoitteena oli selvittää ryhmän toiminnassa esiintyvät ideointitavat, erimielisyydet, ongelmat sekä ryhmän jäsenten aktiivisuus. Koska opetuskokeilun aikana tehokkaaseen havainnointiin jäi vain vähän aikaa, kerätyt havainnot toimivat lähinnä videoiden analysoinnin tukena.

Lukiossa opetuskokeilun havainnointi oli huomattavasti perusastetta helpompaa. Tämän voi selittää sillä, että perusasteella oppilaat ovat herkempiä ärsykkeille. Perusasteella oppilaat tarvitsivat lisäksi huomattavasti enemmän tukea ja ohjausta – lukiolaiset toimivat itsenäisemmin.

6.3 Opetuskokeilun toteuttamissuunnitelma

Syrjälän et al. (1994) mukaan tutkimussuunnitelman laatiminen antaa tutkijalle varmuutta tutkimuksen käynnistämisvaiheessa. Lisäksi sen esittäminen tutkittaville henkilöille herättää luottamusta ja yhteistyöhalua. Heidän mukaansa tutkimussuunnitelma on usein myös edellytyksenä tutkimusluvan saamiselle. Opetuskokeilun aikana emme esittäneet koulujen rehtoreille varsinaista tutkimussuunnitelmaa, vaan esittelimme tutkimuksen kulun ja tavoitteet tuntisuunnitelmien, tutkimusongelmien sekä sanallisien kuvauksien avulla. Kokemattomuutemme opetuskokeilujen tekijöinä voidaan nähdä syynä tutkimussuunnitelman puuttumiseen.

Opetuskokeilu eteni tuntisuunnitelmien mukaan sekä perusasteella (liite 1) että lukiossa (liite 2). Opetuskokeilujen aikana oli tarkoitus kerätä mahdollisimman kattavasti materiaalia ryhmien toiminnasta, vuorovaikutustilanteista sekä Empirica Controlin käytöstä. Materiaalia kerättiin videoimalla ryhmien ongelmanmäärittelyn eteneminen ja Empirica Control -ohjelmistolla mallintamisen vaiheet. Mallintamisvaiheen aikana materiaalia kerättiin myös kannettaviin tietokoneisiin asennettujen ruudunkaappaus-ohjelmistojen avulla. Videoinnin tukena käytimme havainnointia aina, kun kokeilu ei vaatinut täydellistä osallistumistamme toimintaan.

Opetuskokeilu molemmilla kouluilla koostui erillisistä vaiheista, joiden kesto vaihteli käytettävissä olevan ajan mukaan. Ennen varsinaista opetuskokeilua oppilaat täyttivät ennakkokyselylomakkeet (liite 12), joiden avulla jaoimme oppilaat neljän hengen ryhmiin. Oppilaille

selvitettiin aluksi tutkimuksen tarkoitus sekä aineistonhankintatavat. Seuraavassa vaiheessa oppilaille opetettiin ohjelmoinnin perusrakenteet ja heidät tutustutettiin Empirica Control -ohjelmiston käyttöön. Kolmannessa vaiheessa ryhmät määrittelivät ongelman, jonka halusivat ratkaista käyttäen Empirica Control -ohjelmistoa. Ongelmanmäärittelyn jälkeen ryhmät mallinsivat ratkaisun ongelmaansa Empirica Controlilla. Viimeisenä vaiheena opetuskokeilussa oli loppukyselylomakkeiden (liite 13) täyttäminen sekä palautekeskustelu, jossa pyrittiin selvittämään oppilaiden kokemuksia opetuskokeilusta, ohjelmoinnista sekä Empirica Control -ohjelmistosta.

6.4 Kvalitatiivinen lähestymistapa

Opetuskokeilun aikana aineistoa kerättiin kvalitatiivisin menetelmin lähinnä videoinnin, havainnoinnin sekä ruudunkaappausten avulla. Kerätyn videomateriaalin analysoinnissa keskityimme ryhmien toimintaan, varsinkin toiminnassa esiintyviin ongelmatilanteisiin. Analysoidessamme videoita määrittelimme ensin kategoriat, jotka koettiin tärkeiksi ryhmien toiminnan tutkimisen kannalta. Bogdan ja Biklen (1998) toteavat, että varsinkin suuren aineiston kategoriointi koodiryhmiin on tärkeää analysoinnin kannalta. Päädyimme muodostamaan hieman eriävät kategoriat perusastetta (liite 4) ja lukiota (liite 6) varten. Kategorioissa esiintyy erilaista toimintaa spontaaneista reaktioista (naurua) lähtien sekä tapahtumia (kysymys opettajalle) kuvaavia ryhmityksiä. Videoinnin analysoinnissa seurasimme ryhmien vuorovaikutussuhteita. Pyrimme yhdistämään videoilla esiintyvät tilanteet Empirica Control -ohjelmistossa toimimiseen vertaamalla niitä ruudunkaappauksiin.

Havainnointia hyödynsimme opetuskokeilun aikana aktiivisesti lähinnä ongelmanmäärittelyn vaiheessa. Jokaisen opetustilanteen jälkeen kirjoitimme puhtaaksi muistiinpanot, joita tutkimuksen aikana oli kerätty. Bogdanin ja Biklenin (1998) mukaan tutkijan tulisi kirjata ylös mitä tutkimuksen aikana kuullaan, nähdään, koetaan ja ajatellaan. Muistiinpanojen puhtaaksikirjoitus ei tuottanut suuria lisäyksiä kerättyyn aineistoon – tähän vaikutti kokemattomuutemme opetuskokeilujen suorittamisessa. Havainnoinnin osuus väheni huomattavasti mallintamisvaiheessa, sillä siinä vaiheessa vaadittiin huomattavasti enemmän osallistumistamme ryhmien toimintaan.

6.5 Kvantitatiivinen lähestymistapa

Videoiden analysoinnin helpottamiseksi muodostimme materiaaleista kvantitatiiviset aineistot perusasteen (liite 4 & 5) ja lukion (liite 6 & 7) osalta. Erätuulen et al. (1994) mukaan ihmisten asenteita mitattaessa mittayksikön valinta ei ole yksinkertaista. Tutkimuksessa hyödynsimme kvalitatiivisin menetelmin valittuja kategorioita (luku 6.4), joiden avulla kvantitatiivinen aineisto muodostettiin. Videoita analysoidessamme laskimme kullekin kategorialle esiintymiskertojen lukumäärän. Muodostetun numeerisen aineiston avulla tutkimme ryhmien toiminnan yhtäläisyyksiä ja eroja vertailemalla ongelmanmäärittelyn ja Empirica Controlilla mallintamisen vaiheita.

Ennakkokyselyjen (liite 12) perusteella suoritimme ryhmäjaon. Tavoitteenamme oli muodostaa perusasteella kaksi samantasoista ryhmää ja lukiossa kokeneempien ja aloittelevien ohjelmoijien ryhmät. Ennakkokyselyjen analysoinnissa keskityimme tarkastelemaan vastauksia, jotka liittyivät ohjelmointitaitoihin ja tietokoneiden käyttöön sekä ohjelmoinnin perusrakenteiden ymmärtämiseen. Opetuskokeilun jälkeen oppilaat täyttivät loppukyselyt (liite 13), joiden avulla tutkittiin kokeilun vaikutusta ohjelmointiin suhtautumiseen ja ohjelmoinnin perusrakenteiden oppimiseen.

7 OPETUSKOKEILUN KULKU

Tässä luvussa käymme läpi ne vaiheet, joiden kautta opetuskokeilu saatiin suoritettua. Luvussa käsittelemme erillisinä kokonaisuuksina opetuskokeiluja Lieksan lukiolla sekä Merilän perusasteella.

7.1 Opetuskokeilun valmistelu

Opetuskokeilun valmistelu alkoi keväällä 2001, mutta aikataulujen yhteensovittamisessa kohdattujen vaikeuksien vuoksi opetuskokeilu toteutettiin tammi–helmikuussa 2002. Yhteyshenkilöinämme toimivat molempien koulujen rehtorit, joihin pidimme yhteyttä puhelimen sekä sähköpostin välityksellä. Molemmat rehtorit osoittivat kiinnostusta tutkimusta kohtaan sekä tarjosivat mahdollisuuden opetuskokeilun itsenäiseen ja vapaaseen toteutukseen koulujen tiloissa.

Opetuskokeilua varten tiedustelimme kouluilta tarvetta hankkia siihen osallistuvien oppilaiden vanhemmilta lupa kokeilun suorittamiseen. Koulujen rehtorit ilmoittivat, että he voivat valtuuttaa meidät suorittamaan opetuskokeilun koulun tiloissa ja kouluajalla ilman erillisiä lupia oppilaiden vanhemmilta.

Opetuskokeilussa keräsimme materiaalia digitaalivideokameran sekä kahden samanlaisen laitteiston avulla. Laitteistot koostuivat seuraavista komponenteista:

- Kannettava tietokone, hiiri + virtalähde
- Empirica Interface + virtalähde
- Lämpö-, pH-, valaistusvoimakkuus-, massa- ja paineanturi

Lisäksi kannettavissa tietokoneissa oli asennettuna Windows Me-käyttöjärjestelmä, HyperCam-ruudunkaappausohjelmisto sekä Empirica Control -ohjelmiston versio 1.1a.

7.2 Opetuskokeilu Merilän perusasteella *(Harri Kähkönen)*

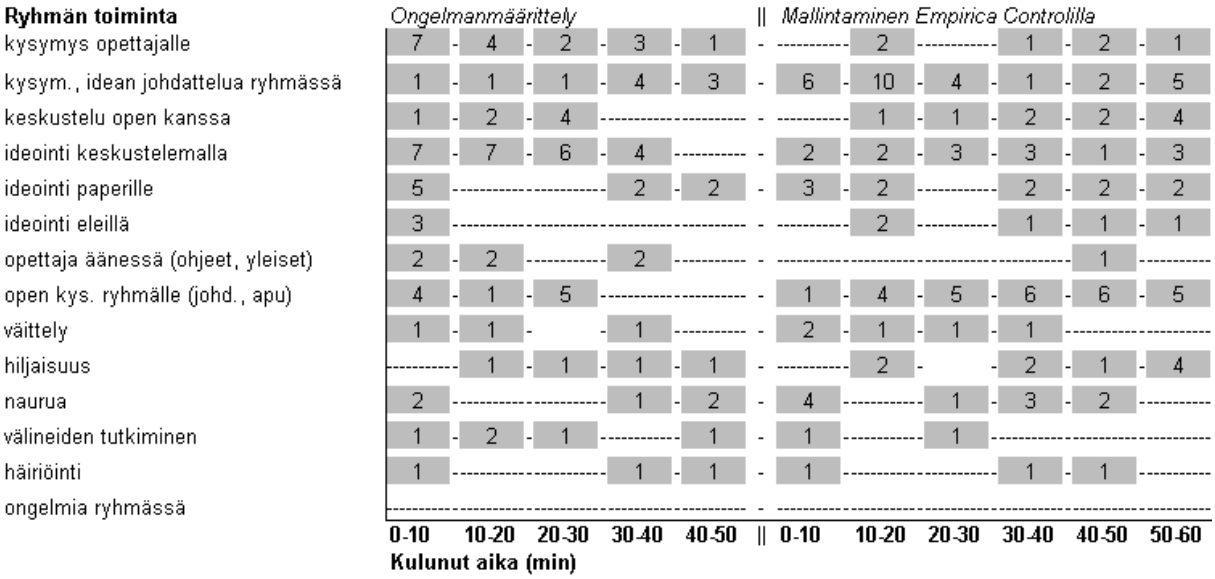
Perusasteen oppilaille toteutimme opetuskokeilun Lieksassa Merilän perusasteella. Opetuskokeiluun osallistui kahdeksan viidennen ja kuudennen luokan oppilasta, joista puolet oli tyttöjä.

Opetuskokeiluun osallistuminen oli vapaaehtoista ja halukkaita olisi ollut enemmän kuin laitteistot ja aika mahdollistivat. Alustavan valinnan tekivät luokkien opettajat. Toiveenamme kuitenkin oli, että oppilailla olisi jonkinlaista kokemusta tietokoneista ennestään, koska rajallinen aika ei tarjonnut mahdollisuutta tarkkaan tietokoneenkäytön opastamiseen. Oppilaat täyttivät ennakkokyselylomakkeen (liite 12), joiden perusteella jaoimme ryhmät. Väärinymmärryksen vuoksi kokeiluun oli alustavasti valittu kymmenen oppilasta vaikka aikataulu ja resurssit antoivat mahdollisuuden vain kahdeksan oppilaan osallistumiselle. Osallistujia valittaessa valintaperusteinamme oli se, että kaikki halukkaat tytöt voivat osallistua kokeiluun. Lisäksi karsimme pojista henkilöt, joilla oli vähiten kokemusta tietokoneiden käytöstä.

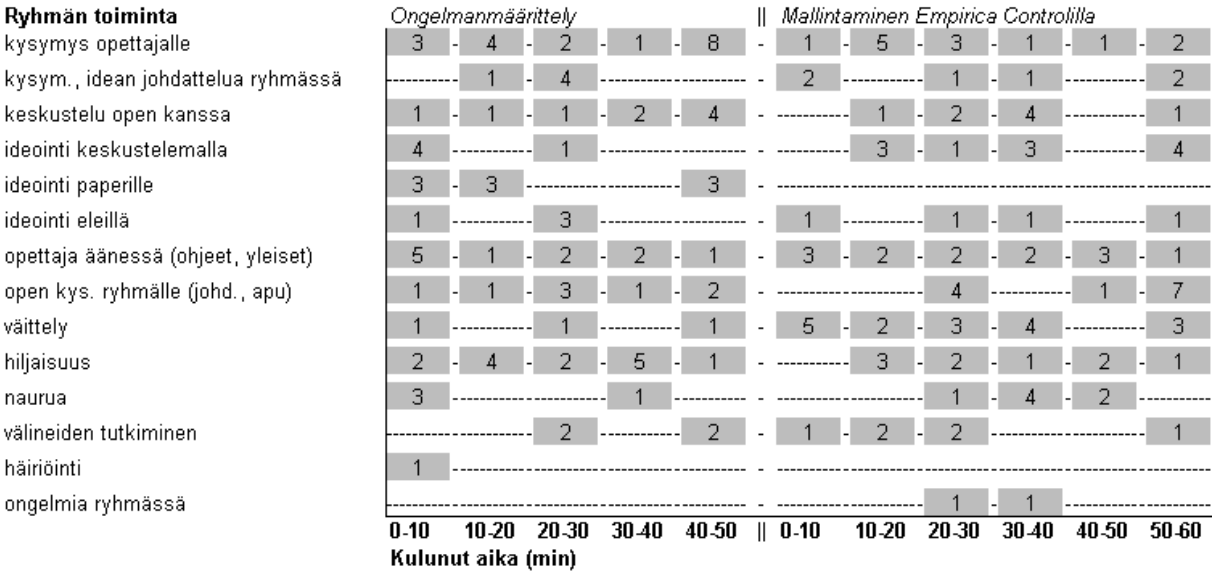
Opetuskokeilun loppuvaiheessa Fruitti-ryhmän tyttö 2 ilmoitti, ettei olisi halunnut osallistua kokeiluun. Emme tiedä miksi oppilas joutui osallistumaan opetuskokeiluun. Luultavasti alustavan valinnan suorittanut opettaja ei saanut tietoa oppilaan haluttomuudesta osallistua. Kuitenkaan kokeilun missään vaiheessa kyseinen oppilas ei osoittanut haluttomuutta osallistua tutkimukseen vaan toimi ryhmässä kuten muutkin jäsenet.

Opetuskokeilu jakautui neljään vaiheeseen. Perusasteella kaksi ensimmäistä tuntia käytettiin ohjelmoinnin perusrakenteiden ja Empirica Control -ohjelmiston käytön opetukseen. Seuraavaksi ryhmät määrittelivät ongelman, jonka he halusivat ratkaista Empirica Control -ohjelmiston avulla. Kolmannessa vaiheessa ryhmät käyttivät Empirica Control -ohjelmistoa ratkaistakseen ongelman. Viimeisenä vaiheena oli palautekeskustelu, jossa oppilailla oli mahdollisuus kommentoida projektin kulkua ja siihen liittyviä asioita. Lisäksi oppilaat täyttivät loppukyselyn (liite 13).

Ryhmien toimintaa kokeilun eri vaiheissa kuvattiin aikajanoilla, jotka muodostettiin havainnoinnin ja videoiden analysoinnin perusteella. Aikajanoissa on kuvattu esiintyvä toiminta tietyllä ajanjaksolla, sekä toiminnan esiintymiskertojen lukumäärä numeroarvona. Sekä Aprikoosi-ryhmällä (kuva 12) että Fruitti-ryhmällä (kuva 13) toiminta eteni saman aikataulun mukaisesti.



Kuva 12: Aprikoosi-ryhmän toiminta opetuskokeilun aikana.



Kuva 13: Fruitti-ryhmän toiminta opetuskokeilun aikana.

7.2.1 Ennakkokyselyjen vaikutukset

Ennen opetuskokeilua toteutettujen ennakkokyselyiden (liite 12) perusteella suoritettiin ryhmäjako. Ryhmät jaettiin ennakkokyselyjen perusteella tiedoiltaan ja taidoiltaan mahdollisimman tasapainoisiin ryhmiin. Molempiin ryhmiin pyrittiin saamaan aiempaa kokemusta tietotekniikasta samassa suhteessa. Oppilaat, joilla oli aiempaa ohjelmointikokemusta jaettiin tasapuolisesti molempiin ryhmiin. Käytännössä tämä tarkoitti sitä, että molemmissa ryhmissä oli kaksi poikaa ja kaksi tyttöä. Pojilla oli kyselyjen perusteella huomattavasti laajemmat koke-

mukset tietotekniikasta tyttöihin verrattuna. Ryhmät saivat itse päättää ryhmälleen nimen. Oppilaat nimesivät ryhmät Aprikoosiksi ja Fruitiksi.

7.2.2 Ohjelmoinnin perusrakenteet

Ohjelmoinnin perusrakenteiden, ehto- ja toistolauseen sekä muuttujan opettaminen perusasteikäisille toteutettiin keskustelemalla ja esimerkkien avulla. Opetus eteni tuntisuunnitelman (liite 1) mukaisesti. Pyrimme vapaan keskustelun avulla herättämään oppilaiden keskuudessa ideoita ja ehdotuksia, joiden avulla ohjelmointirakenteita voitaisiin käsitellä. Ohjelmointi käsitteenä oli lähes kaikille oppilaille vieras, joten perusasiat tuli käydä läpi huolellisesti. Ohjelmoinnin perusrakenteet pyrittiin sitomaan konkreettiseen maailmaan, josta poimittiin esimerkkejä kuten kahvinkeittoalgoritmi. Esimerkit käytiin yhdessä läpi taululla käyttäen Empirica Control -ohjelmiston ohjelmointirakenteita kuvaavia kuvakkeita apuna. Oppilaille esitettiin jatkuvasti kysymyksiä, joiden tarkoituksena oli aktivoida heitä esittämään ideoita ja kysymyksiä esimerkkien muokkaamiseksi ja selventämiseksi. Keskusteluvaiheessa havaitsimme että ryhmien pojat, joita ohjelmointi ja tietotekniikka kiinnosti eniten, olivat aktiivisimmin äänessä.

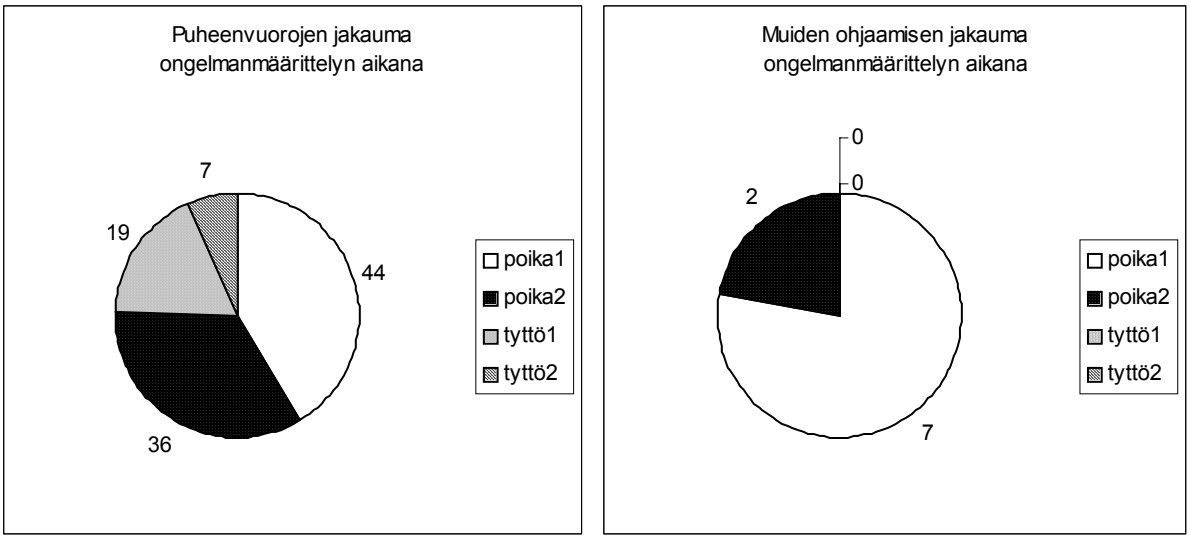
Ohjelmoinnin perusrakenteita käsiteltiin rajallisen ajan vuoksi vain tunti, jonka jälkeen siirryttiin kokeilemaan niiden toimintaa käytännössä Empirica Control -ohjelmistolla. Ryhmät ratkaisivat yhdessä antamiamme tehtäviä. Niissä edettiin yksinkertaisista ohjelmarakenteista, kuten jatkuva lämpötilan mittaaminen, kohti monimutkaisempaa kokonaisuutta. Oppilaita pyydettiin kehittämään lämpötilaa tarkkailevaa ohjelmaa lisäämällä siihen ehtoja sekä loogisia operaatioita. Tehtävissä pyrittiin käyttämään ohjelmoinnin perusrakenteita mahdollisimman monipuolisesti. Niissä harjoiteltiin myös lokikirjan ja äänten käyttöä.

Opetuskokeilun suunnittelussa ohjelmoinnin perusrakenteiden opettamiseen oli varattu aikaa noin kaksi tuntia. Vaikka perusasteella ohjelmointi oli käsitteenä vieras, aikataulu ei antanut mahdollisuutta perusteellisempaan opetukseen. Ajan puute näkyi varsinkin muuttujan käytön opettamisessa. Muuttujaa käsitteenä ei ilmeisesti omaksuttu, vaikka sitä hyödyntäviä esimerkkejä käytiin läpi useita. Myöhemmin mallintamisvaiheessa muuttujasta puhuttiin ja sitä käytettiin avustuksellamme. Kuitenkin loppukyselyt osoittivat, etteivät oppilaat täysin omaksuneet muuttujan käyttömahdollisuuksia.

7.2.3 Brainstorming – ongelman määritteleminen

Ongelman määrittelemisessä oppilaille annettiin tehtäväksi kehittää ongelma, jonka he voisivat Empirica Control -ohjelmiston avulla mallintaa. Mitään suljettua ongelmaa ei esitetty, vaan ryhmät saivat vapaasti pohtia eri mahdollisuuksia ja ehdottaa niitä toteutettaviksi. Oppilaille annettiin tehtäväksi ainoastaan kirjata mahdolliset ideat ja ehdotukset paperille ja kehittää ongelmaa eteenpäin esimerkiksi suunnittelemalla ohjelman suoritusta. Ongelman määritteleminen onnistui molemmilta perusasteen ryhmiltä yllättävän hyvin verrattuna lukion vastaavaan prosessiin. Oppilaat pääsivät nopeasti yhteisymmärrykseen ongelmasta, vaikkakin ryhmän sisäiset suhteet tulivat nopeasti selville. Osa oppilaista oli selvästi toisia aktiivisempia (kuva 14).

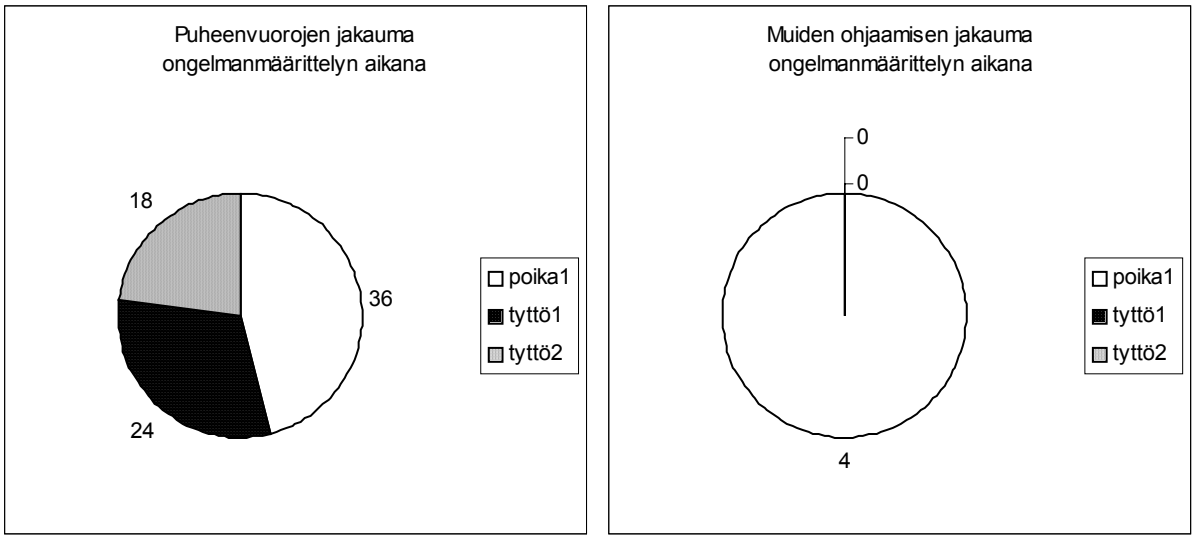
Aprikoosin keskusteluissa pojat ottivat selkeät roolit ideoijina ja keskustelun ylläpitäjinä. He käyttivät ongelmanmäärittelyprosessin aikana noin kolme neljäsosaa kaikista puheenvuoroista. Tyttöjen ideat eivät saaneet niille kuuluvaa huomiota, sillä pojat puolustivat voimakkaasti omia ideoitaan. Poikien aktiivisempi käyttäytyminen ilmeni myös siten, että he neuvoivat tyttöjä huomattavasti enemmän (kuva 14). Kuitenkin ryhmä keskusteli ja myös ideoi yhdessä. Muutaman kerran ryhmän jäsenet jopa väittelivät keskenään. Osallistuimme ryhmän toimintaan lähinnä antaen virikkeitä kysymällä ja vastaamalla ryhmän kysymyksiin. Emme antaneet kuitenkaan suoria vastauksia. Tavoitteenamme oli johdatella ryhmää itsenäiseen ajatteluun ja ongelmanratkaisuun. Ryhmän meille esittämät kysymykset koskivat lähinnä Empirica-mittausjärjestelmän fyysisistä laitteistoa sekä ongelman määrittelemistä paperille.



Kuva 14: Aprikoosin ongelmanmäärittely.

Aprikoosi-ryhmän ongelmanmäärittelyn tuloksena oli varashälytin. Heidän suunnitelmansa oli toteuttaa Empirica Control -ohjelmistolla varashälytin, joka tarkkailisi huonetta ja hälyttäisi, jos varas saapuisi huoneeseen ikkunasta tai pääsisi kassakaapille asti. Ryhmä suunnitteli käyttävänsä mallintamisessa Empirica-mittausjärjestelmän valaistusvoimakkuus- sekä lämpötila-antureita.

Fruitti-ryhmän kokoonpano muuttui kesken ongelman määrittelyn, koska ryhmän poika 2 sairastui ja oli poissa opetuskokeilun toisena päivänä. Emme analysoineet kyseisen oppilaan toimintaa videoita katsoessamme, joten Fruitti-ryhmästä muodostui kolmihenkinen. Poistettu oppilas oli jo ensimmäisenä tutkimuspäivänä erittäin hiljainen ja osallistui ryhmän toimintaan lähinnä seuraajana. Fruitti-ryhmässä puheenvuorot jakaantuivat selvästi tasaisemmin kolmen oppilaan kesken. Kuitenkin ryhmän ainoa poika oli selvästi kiinnostunein tehtävästä ja hän käytti eniten puheenvuoroja. Ryhmän poika myös neuvoi tyttöjä ongelmanmäärittelyvaiheessa (kuva 15). Ryhmässä esiintyi myös väittelyä ja lisäksi toista ryhmää huomattavasti enemmän hiljaisuutta. Ryhmä käytti myös huomattavasti enemmän apuamme toiminnassaan kysellen neuvoja ja keskustellen kanssamme.



Kuva 15: Fruitin ongelmanmäärittely.

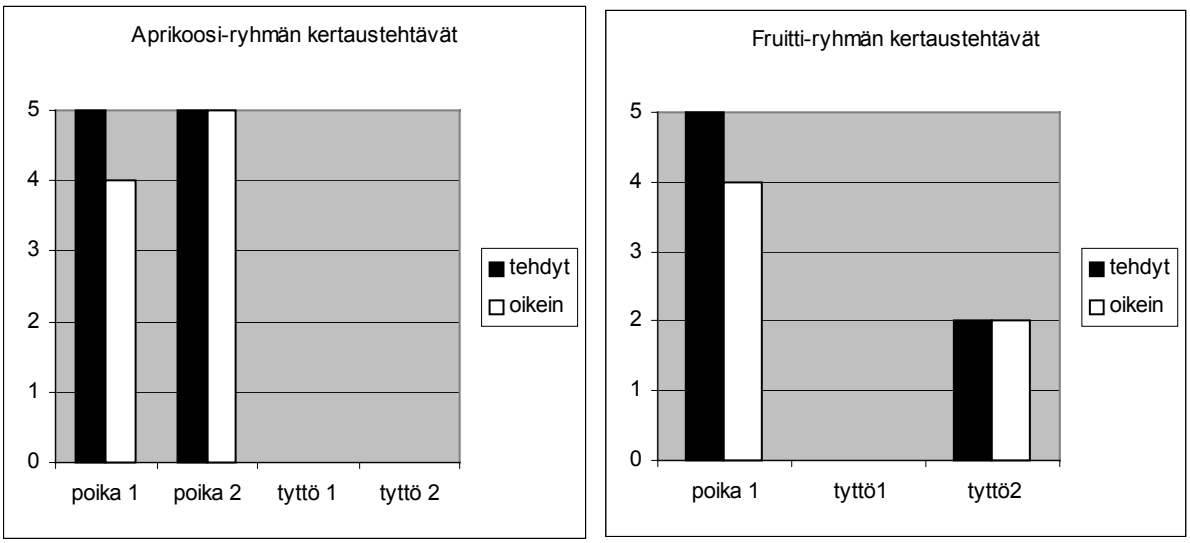
Fruitti-ryhmän ongelmanmäärittelyn tuloksena oli kasvihuoneessa vallitsevia olosuhteita tarkkaileva järjestelmä. Ryhmän suunnitelman mukaan Empirica Control -ohjelmistolla toteutettu järjestelmä varoittaisi, jos valaistusvoimakkuus- tai lämpötila-antureiden mittaamat arvot ylittäisivät tai alittaisivat sallitut rajat.

7.2.4 Kertaustehtävät

Koska opetuskokeilu suoritettiin kahtena peräkkäisenä keskiviikkona, katsoimme tarpeelliseksi laatia oppilaille harjoituksia, joiden avulla Empirica Control palautuisi helpommin heidän mieleensä. Ennen jälkimmäistä opetuskokeilupäivää oppilailla oli kaksi päivää aikaa tehdä kertausharjoituksia (liite 3). Tehtävät eivät olleet pakollisia vaikkakin niitä suositeltiin tehtäviksi. Tehtävissä oppilaiden piti selittää miten valmiit ohjelmat toimisivat. Kertaustehtävät käytiin läpi yhdessä keskustellen toisen testauspäivän aamuna ennen muun toiminnan aloittamista.

Molemmissa ryhmässä pojat olivat tehneet kaikki kertaustehtävät, vaikkakin viimeinen tehtävä tuotti ongelmia. Tytöistä vain yksi oli tehnyt osan kertaustehtävistä (kuva 16). Kertaustehtävät etenivät yksinkertaisemmista monimutkaisempiin, koska tavoitteena oli ettei kukaan säikähtäisi tehtävien vaikeutta heti alkuunsa. Vaikka viidennen kertausharjoituksen laskuria ei ehditty oppitunnilla käydä läpi, molemmista ryhmistä yksi poika oli ymmärtänyt laskurin käyttämisen oikein. Fruitti-ryhmän pojalle viive-kuvake tuotti kertaustehtävissä ongelmia. Hän ymmärsi kuvakkeen määräävän koko ohjelman suoritukseen kuluvan ajan.

Mallintamisprosessin päätyttyä kaksi tyttöä, joilla kertaustehtävät olivat jääneet tekemättä, halusivat yrittää ratkaista niitä. Tulokset jäivät kuitenkin heikoiksi osittain ajanpuutteen takia. Lisäksi molempien tyttöjen vastaukset ensimmäisiin kertaustehtäviin olivat identtiset.



Kuva 16: Kertaustehtävät perusasteella.

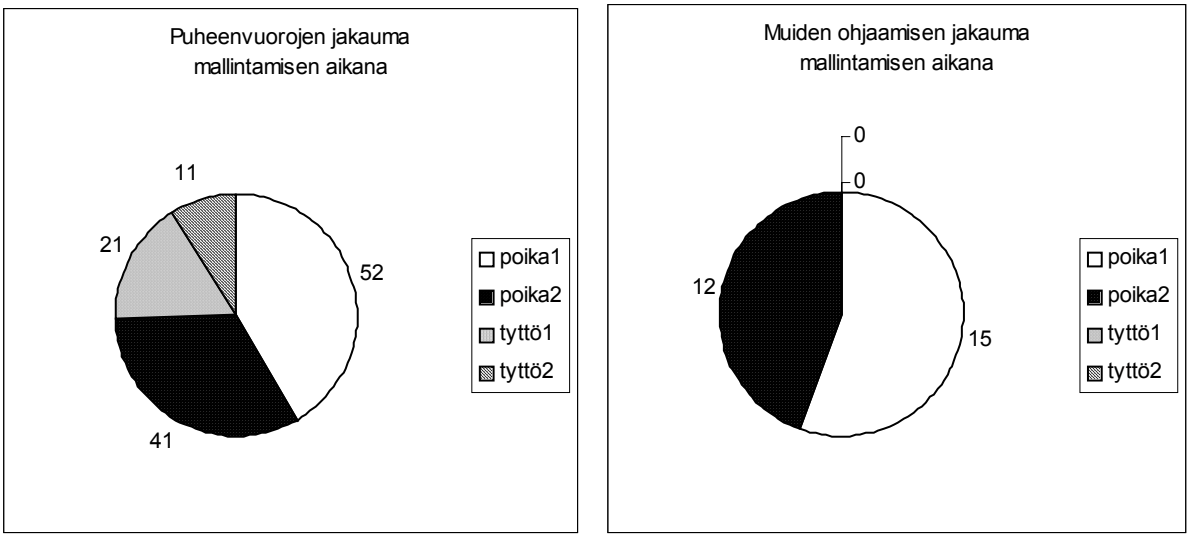
7.2.5 Empirica Contol -mallintaminen

Mallintamisvaiheessa ryhmien toiminta oli huomattavasti aktiivisempaa kuin ongelmanmäärittelyvaiheessa. Ryhmät keskustelivat huomattavasti aktiivisemmin ja vapautuneemmin aiempaan verrattuna. Roolimme oli edelleenkin ryhmien toimintaa aktivoiva. Ratkaisujen sijaan esitimme kysymyksiä, joiden tarkoituksena oli saada ryhmät pohtimaan sekä kehittämään mallinsa toimintaa. Ryhmien sisäiset suhteet pysyivät jokseenkin samoina. Pojat olivat edelleen aktiivisempia.

Aprikoosin toimintaa hallitsivat edelleen pojat käyttäen suurimman osan puheenvuoroista (kuva 17). Pojat neuvoivat tyttöjä mallintamisen edetessä usein ja ryhmässä esiintyi keskustelua, väittelyä ja naurua huomattavasti enemmän kuin ongelmanmäärittelyvaiheessa. Ryhmässä kaikki olivat koneen ääressä ja seurasivat mallintamisen etenemistä. Kukaan ei poistunut koneen äärestä kesken toiminnan.

Aprikoosi sai rakennettua ohjelmansa ilman suuria ongelmia. Aiemmat suunnitelmat, jotka he olivat laatineet, palvelivat mallinnustilannetta hyvin ja ohjelma syntyi nopeasti. Ryhmän mallintamisessa oli lisäksi huomionarvoista se, että loogiset operaatiot syntyivät ongelmitta. Ai-

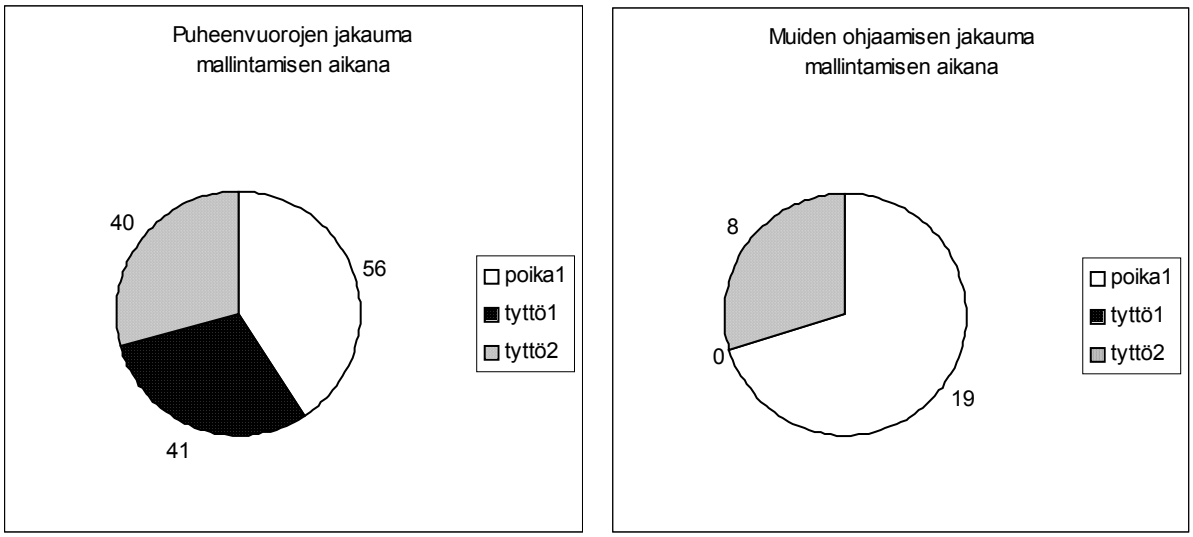
noastaan kahden mittauksen loogisen operaation muodostamisessa oli pieniä hankaluuksia. Ryhmän pojat kuitenkin korjasivat ohjelmarakennetta omatoimisesti niin, että se huomioi eri mittaukset. Muuttujat asetettiin ohjelmaan avustuksellamme. Muuttuja käsitteenä oli ryhmällä tiedossa, mutta sen soveltaminen tuotti ongelmia.



Kuva 17: Aprikoosin mallintaminen.

Aprikoosi-ryhmän toiminnasta huomasi selkeästi, että ensimmäinen ratkaisu ongelmaan tyydytti ryhmää liiankin hyvin. Johdatteluistamme huolimatta mallin kehitteleminen ja pohtiminen ei ryhmää kiinnostanut. Ryhmän valmis malli eli varashälytin (liite 8) saatiin valmiiksi ilman suurempia ongelmia. Ryhmän toimintaa ohjasivat vahvasti pojat, mutta tytötkin osallistuivat mallintamiseen tietokoneen käyttäjinä sekä ideoijina.

Fruitin mallintamisvaiheessa pojan ja tyttöjen välisen kommunikaation ongelmat ilmenivät selvemmin. Ryhmän poika oli aktiivisempi suunnittelemaan ja kehittelemään mallia, kun taas tytöt olivat enemmän kiinnostuneita ohjelman muista ominaisuuksista, kuten äänien testailusta. Ryhmän sisällä käytiin välillä äänekkäitäkin väittelyitä ja pojan turhautuminen kävi ilmi useampaan kertaan. Ryhmän jäsenet halusivat selvästi eri asioita. Ryhmän poika keskusteli aktiivisesti kanssamme osoittaen samalla kiinnostuksensa Empirica Control -ohjelmistolla toimimista kohtaan. Hän esitti meille huomioitaan ja ideoitaan aktiivisesti. Ongelmista huolimatta ryhmä teki töitä yhdessä ja keskustelu oli vilkkaampaa kuin ongelmanmäärittelyvaiheessa (kuva 18).



Kuva 18: Fruitin mallintaminen.

Fruitti-ryhmässä ilmeni selkeämmin ongelmia ohjelmarakenteiden luomisessa. Loogisen operaation saaminen ohjelmaan osoittautui ylivoimaiseksi, joten jouduimme opastamaan sen tekemisessä kädestä pitäen. Ruudunkaappauksista oli havaittavissa, että ryhmä oli ymmärtänyt loogisten operaatioiden luomisen, mutta käytännössä tehtävä osoittautui hankalaksi. Hiiren kaksoisnapautukset eksyivät kuvakkeiden ulkopuolelle, jolloin Empirica Control ei niihin reagoinut. Myös lokikirjan käytössä ilmeni ongelmia, kun ryhmä ei huomannut asettaa lokikirjoille omaa suoraa mittaustuloa. Ryhmä unohti myöskin taarata anturit ennen ohjelman suoritusta, mutta virhe huomattiin ja korjattiin välittömästi ohjelman ajon jälkeen. Fruitti-ryhmäläisillä oli ongelmia myös muuttujien luomisen kanssa, joten opastimme ryhmää niiden laatimisessa.

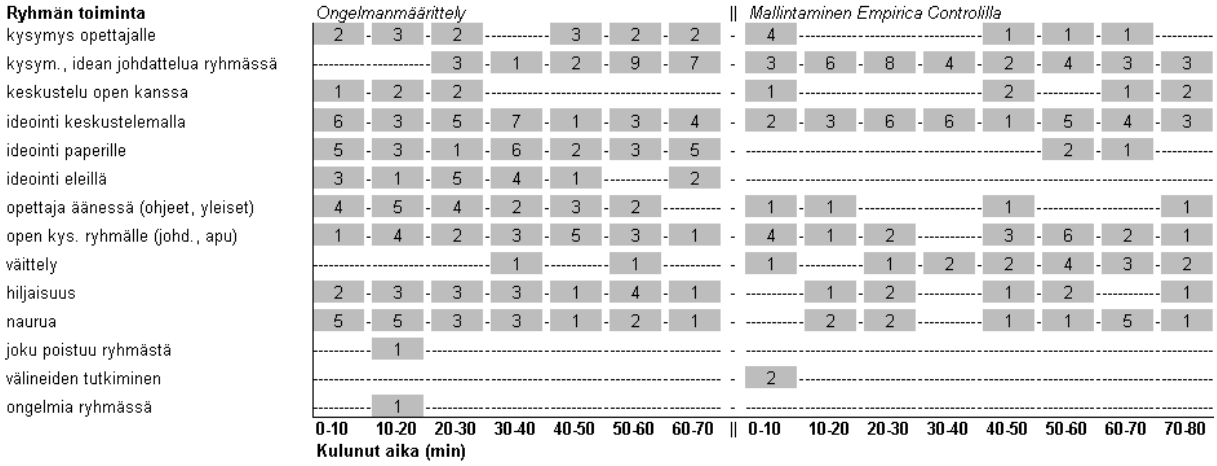
Ryhmän valmis ohjelma (liite 9) tarkkaili kasvihuoneen olosuhteita. Kun pyysimme ryhmää selvittämään, kuinka ohjelma toimii, havaittiin selvästi tyttöjen rooli enemmän sivustakatsojina. Tytöt kertoivat itse, etteivät tajunneet ohjelman toimintaa. Suhtaudumme näihin kommentteihin kuitenkin varauksella, koska tytöt osallistuivat ohjelmointiin siinä missä ryhmän poikakin. Selvää oli ainoastaan se, että ryhmän poika oli selvästi tyttöjä enemmän kiinnostunut Empirica Controlilla mallintamisesta. Mallintamisen aikana neuvoja antoi myös ryhmän tyttö 2 (kuva 18).

7.3 Opetuskokeilu Lieksan lukiolla *(Kai Piiroinen)*

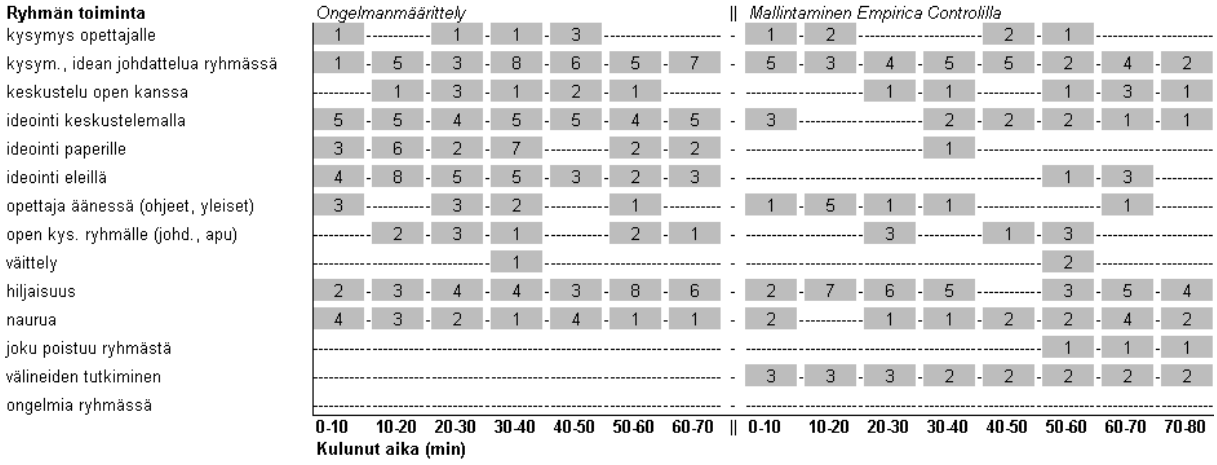
Opetuskokeilu toteutettiin tammikuussa 2002 Lieksan lukiossa kahtena peräkkäisenä päivänä joiden aikana käytössämme oli kuusi oppituntia (3+3). Opetuskokeiluun osallistui kahdeksan lukion ensimmäisen vuosikurssin oppilasta, joista viisi oli tyttöjä. Kokeiluun osallistuminen oli vapaaehtoista ja se toteutettiin meneillä olleen tietotekniikkakurssin aikana. Kokeiluun osallistuneet oppilaat valitsi alustavasti Lieksan lukion tietotekniikanopettaja. Toiveenamme oli, että osallistuneilla olisi mahdollisimman vähän ohjelmointikokemusta. Oppilaat täyttivät lisäksi ennakkokyselylomakkeen (liite 12). Jaoimme oppilaat kahteen ryhmään heidän vastaustensa perusteella. Kriteerinä oli aiempi tietotekniikan tuntemus, erityisesti oppilaiden ohjelmointitaidot. Tehdessämme lopullisen ryhmäjaon pyrimme jakamaan aiempaa ohjelmointikokemusta omaavat oppilaat omaan ryhmäänsä. Oletimme, että näin ryhmissä voitaisiin saavuttaa erilaisia vuorovaikutustilanteita sekä lähestymistapoja ratkaistaviin ongelmiin. Kokeneempien ohjelmoijien ryhmään valitsimme kaksi poikaa ja tyttöä, joista kolmella oli ohjelmointikokemusta Java-kielellä, kun taas Aloittelevien ryhmään valittiin yksi poika ja kolme tyttöä.

Opetuskokeilu jakautui neljään vaiheeseen. Lukiolla opetuskokeilun ensimmäinen tunti käytettiin ohjelmoinnin perusrakenteiden sekä Empirica Control -ohjelmiston opetukseen. Seuraavaksi ryhmät määrittelivät ongelman, jonka he halusivat ratkaista. Kolmannessa vaiheessa ryhmät ratkaisivat luomansa ongelman Empirica Control -ohjelmiston avulla. Viimeisenä vaiheena opetuskokeilussa oli palautekeskustelu, jossa oppilailla oli mahdollisuus kommentoida kokeilun kulkua ja siihen liittyviä asioita. Lisäksi oppilaat täyttivät loppukyselylomakkeen (liite 13).

Ryhmien toimintaa opetuskokeilun ongelmanmäärittelyn- ja mallintamisvaiheissa kuvattiin aikajanoilla, jotka muodostettiin havainnoinnin ja videoiden analysoinnin perusteella. Aikajanoissa on kuvattu esiintyvä toiminta tietyllä ajanjaksolla sekä esiintymien lukumäärä numeroarvona. Sekä Aloittelevien ryhmän (kuva 19) että Kokeneiden ryhmän (kuva 20) työskentely eteni saman aikataulun mukaisesti.



Kuva 19: Aloittelevien ryhmän toiminta opetuskokeilun aikana.



Kuva 20: Kokeneiden ryhmän toiminta opetuskokeilun aikana.

7.3.1 Ennakkokyselyjen vaikutukset

Ennen opetuskokeilua täytettyjen ennakkokyselyiden (liite 12) perusteella suoritettiin ryhmäjako. Ryhmät jaettiin kokeneisiin ja aloitteleviin ohjelmoijiin tavoitteena luoda selkeä vastakkainasettelu ryhmien välille. Ryhmät nimettiin jäsenten ohjelmointitason mukaisesti Kokeneiden ja Aloittelevien ryhmiksi. Aloittelevien ryhmään valitsimme kaikki ne oppilaat, joilla ei ollut aiempaa ohjelmointikokemusta. Kokeneempien ryhmään valitsimme ne oppilaat jotka olivat käyneet ohjelmointikursseja tai olivat muuten kokeneempia tietokoneen käyttäjiä. Oppilaat huomasivat tavoitteemme välittömästi, kun ilmoitimme ryhmäjaon. Tämän suurempaa keskustelua ei asiasta kuitenkaan syntynyt ja oppilaat hyväksyivät tekemämme ryhmäjaon.

7.3.2 Ohjelmoinnin perusrakenteet

Lukiossa ohjelmoinnin perusrakenteiden, ehto- ja toistolauseen sekä muuttujan opettaminen toteutettiin keskustelemalla ja esimerkkien avulla. Opetus eteni tuntisuunnitelman (liite 2) mukaisesti. Pyrimme vapaasti keskustellen herättämään sellaisia ideoita ja ehdotuksia oppilaiden keskuudessa, joiden avulla ohjelmointirakenteita voitaisiin käsitellä. Osalle oppilaista oli muodostunut kuva ohjelmoinnin perusrakenteista jo aiemmilla ohjelmointikursseilla. Keskustelun aikana pyrimme esimerkein kuvaamaan ohjelmarakenteita sekä sitomaan niitä konkreettiseen maailmaan. Yhtenä esimerkkinä käytimme kahvinkeittoalgoritmia. Oppilaita pyrittiin aktivoimaan keskusteluun esittämällä kysymyksiä sekä pyytämällä esimerkkejä ohjelmointirakenteiden käytöstä esimerkiksi omassa toiminnassaan.

Lukiossa ohjelmoinnin perusrakenteiden opettamiseen oli aikaa vielä vähemmän kuin perusasteella. Keskustelun aikana oppilaat vastailivat kysymyksiin aktiivisesti ja vaikutti siltä, että he omaksuivat ohjelmoinnin perusrakenteet hyvin. Tästä huolimatta esimerkiksi muuttujan määritelmä ja toiminta jäi monille oppilaille avoimeksi kuten perusasteellakin. Loppukyselyistä selvisi, että monille oppilaille juuri muuttujan käyttötarkoitukset ja määritelmä tuottivat ongelmia.

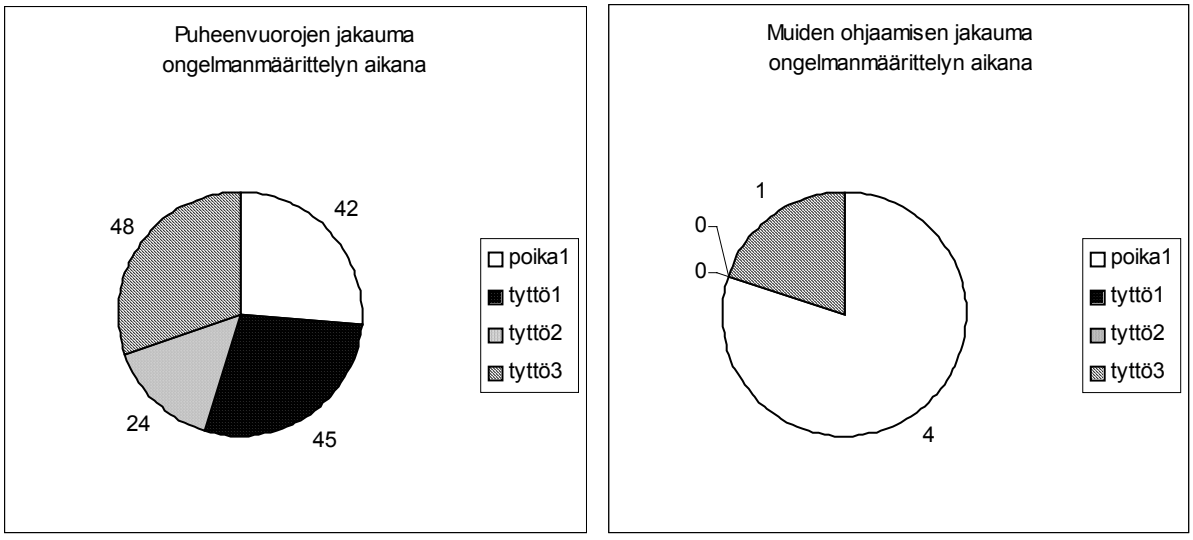
Lukiossa ohjelmoinnin perusrakenteet yhdistettiin Empirica Control -ohjelmistoon vasta, kun siihen tutustuttiin. Tarkoituksenamme oli opastaa ryhmiä siten, että antamiemme esimerkkien avulla ryhmät kykenisivät yhdistämään ohjelmoinnin perusrakenteet Empirica Control -ohjelmiston vastaaviin kuvakkeisiin. Rajallinen aika ei antanut mahdollisuutta Empirica Controlin kattavaan kokeiluun ja suuri vastuu sen omaksumisesta jäi ryhmille. Opetuskokeilu osoitti, että oppilaat omaksuivat nopeasti ja vaivattomasti Empirica Control -ohjelmiston graafisen käyttöliittymän ja ohjelmointikielen.

7.3.3 Brainstorming – ongelmanmääritteleminen

Ongelmanmäärittelemisessä ryhmille annettiin tehtäväksi luoda ongelma, jonka he voisivat Empirica Control -ohjelmiston avulla mallintaa. Mitään suljettua ongelmaa ei esitetty, vaan ryhmät saivat vapaasti pohtia eri mahdollisuuksia ja ehdottaa niitä toteutettaviksi. Ryhmille annettiin tehtäväksi ainoastaan kirjata mahdolliset ideat ja ehdotukset paperille ja kehittää niitä eteenpäin esimerkiksi suunnittelemalla ohjelman suoritusta. Suurimmat ongelmat luki-

ossa syntyivät lähinnä ongelmanmäärittelyn aikana. Jostain syystä ongelman määrittely ja brainstorming -menetelmänä tuottivat suuria ongelmia varsinkin Aloittelevien ryhmälle, jonka jäsenillä ei entuudestaan ollut ohjelmointikokemuksia. Ilmeisesti koko brainstorming-metodi oli oppilaille melko tuntematon. Kun ongelmaa ryhdyttiin ryhmissä määrittelemään, Aloittelevien ryhmällä ei heti ollut selkeää kuvaa siitä, kuinka heidän tulisi toimia. Tähän saattoi vaikuttaa oppilaiden kokemattomuus ohjelmoinnin osalta. Heille ei ollut vielä muodostunut kuvaa siitä, mitä Empirica Control -ohjelmistolla on mahdollista tehdä. Aloittelevien ryhmän toiminta oli kuitenkin erittäin aktiivista ja ryhmän ilmapiiri oli positiivinen. Ryhmän kaikki jäsenet käyttivät paljon puheenvuoroja ja ryhmäläiset myös neuvoivat toisiaan ongelmatilanteissa (kuva 21).

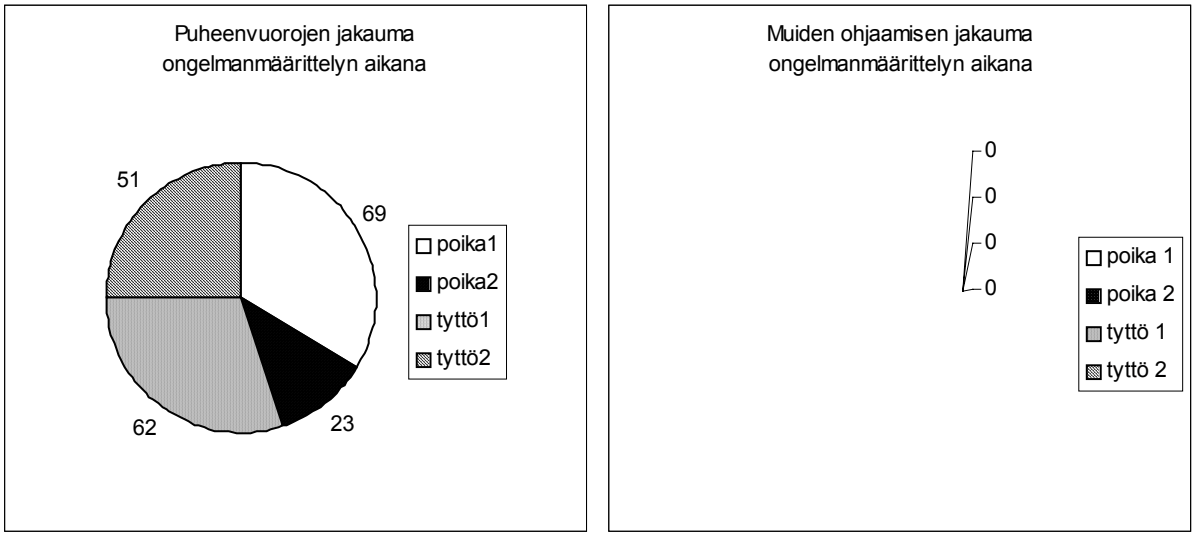
Ensimmäisenä opetuskokeilupäivänä Aloittelevien ryhmässä oli myös havaittavissa pientä turhautumista, kun toteuttamiskelpoista ideaa ei keksitty hetkessä. Jotta ryhmän toiminta saataisiin liikkeelle, meidän oli pakko vastailla kysymyksiin vastakysymyksillä. Pyrimme näin aktivoimaan oppilaita ajattelemaan ja pohdiskelemaan uusilla tavoilla. Aloittelevien ryhmässä toteuttamiskelpoisia ideoita kyllä esiintyi, mutta niitä ei jalostettu riittävän pitkälle, eivätkä ne siten saaneet kovin suurta kannatusta. Ryhmän toiminta oli ongelmista huolimatta erittäin aktiivista ja selvästikin ryhmän jäsenet tekivät parhaansa toteutuskelpoisen idean löytämiseksi.



Kuva 21: Aloittelevien ryhmän ongelmanmäärittely.

Pelko siitä, että ryhmä ei keksisi toteuttamiskelpoista ideaa osoittautui turhaksi. Toisen opetuskokeilupäivän alussa kävi ilmi, että Aloittelevien ryhmän jäsenet olivat pohtineet ongelmia kotonaan kysellen vanhemmiltaan esimerkiksi auton polttoaineen kulutukseen vaikuttavista tekijöistä. Ryhmä keksi kuitenkin uuden ongelman, joka sai kannatusta ryhmän keskuudessa. Ryhmä päätti mallintaa Empirica Control -ohjelmiston avulla liskon terraarion elinolosuhteita tarkkailevan järjestelmän käyttäen valaistusvoimakkuus- ja lämpötila-anturia.

Kokeneempien ryhmässä oli oppilaita, joilla oli entuudestaan kokemuksia ohjelmoinnista, lähinnä Java-ohjelmointikielestä. Ryhmän ongelmanmäärittely sujui huomattavasti vaivattomammin kuin Aloittelevien ryhmän. Ongelmanmäärittely onnistui jo ensimmäisen opetuskokeilupäivän aikana. Ryhmä laati paperille ratkaisun, joka oli lähes valmis Empirica Controlilla toteutettava ohjelma. Kokeneiden ryhmän toiminnassa poika 2 jäi taka-alalle, enemmänkin tarkkailijaksi. Huomioitavaa on kuitenkin, ettei hän eristäytynyt kokonaan, vaan osallistui ajoittain ryhmän toimintaan ja ideointiin. Ongelmanmäärittelyn aikana Kokeneiden ryhmässä ei esiintynyt neuvontaa, vaan ryhmän jäsenet keskustelivat yhdessä mahdollisesta ongelman ratkaisukeinosta (kuva 22). Kokeneiden ryhmän työskentely oli koko ongelmanmäärittelyn ajan aktiivista ja tehokasta.



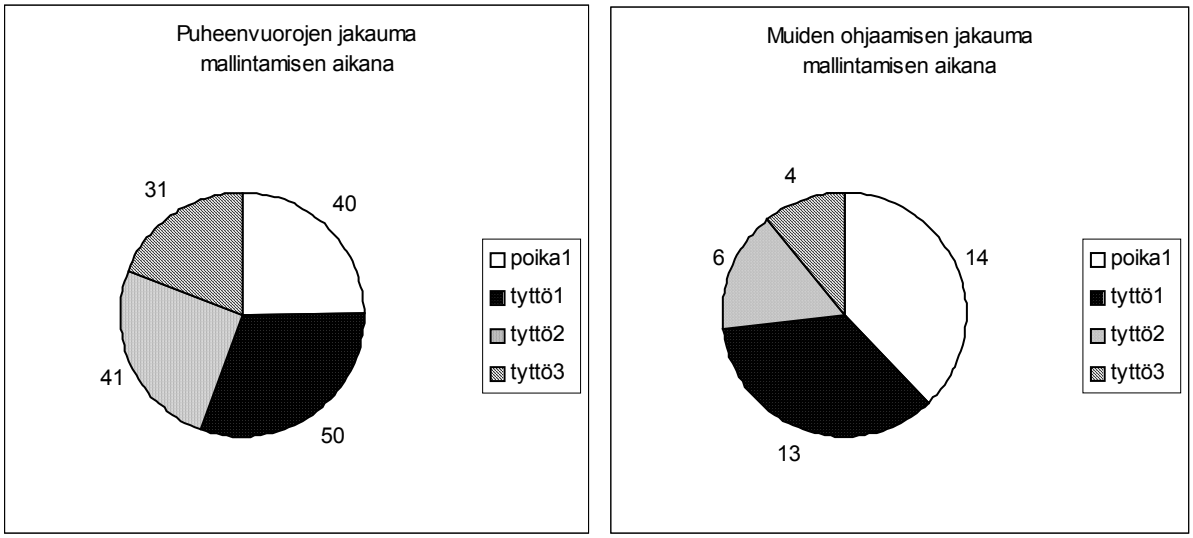
Kuva 22: Kokeneempien ryhmän ongelmanmäärittely.

Kokeneempien ryhmän nopea ongelmanmäärittely ei johtanut lisätutkimuksiin tai ongelman jalostamiseen. Muita vaihtoehtoisia ongelmia ei edes pohdittu yhteisen ongelman hyväksymisen jälkeen. Kokeneempien ryhmän tavoitteena oli tutkia kuinka valo läpäisee ilmapallon ku-

min, kun ilmapalloa venytetään. Massa-anturin avulla ryhmä tutki kuinka voimakkaasti ilmapalloa venytetään ja valaistusvoimakkuus-anturin avulla ryhmä mittasi valon läpäisevyyttä.

7.3.4 Empirica Control -mallintaminen

Varsinainen mallintamisprosessi sujui molemmilta ryhmiltä hyvin. Oma roolimme oli edelleenkin ryhmien toimintaa aktivoiva. Ratkaisujen sijaan esitimme kysymyksiä, joiden tarkoituksena oli saada ryhmät pohtimaan ja kehittämään mallinsa toimintaa. Aloittelevien ryhmän ainoa poika ja tyttö 1 olivat muita aktiivisempia. Mallintamisvaiheen aikana ryhmässä käytiin varsin paljon väittelyjä. Etenkin ryhmän poika ja tyttö 1 keskustelivat toteutustavoista muutaman kerran melko äänekkäästi. Kyseiset henkilöt olivat lisäksi aktiivisimpia neuvojia ja käyttivät tietokonetta ja muita välineitä enemmän kuin ryhmän muut jäsenet (kuva 23). Aloittelevien ryhmässä keskustelu säilytti asemansa vallitsevana ideointikeinona.



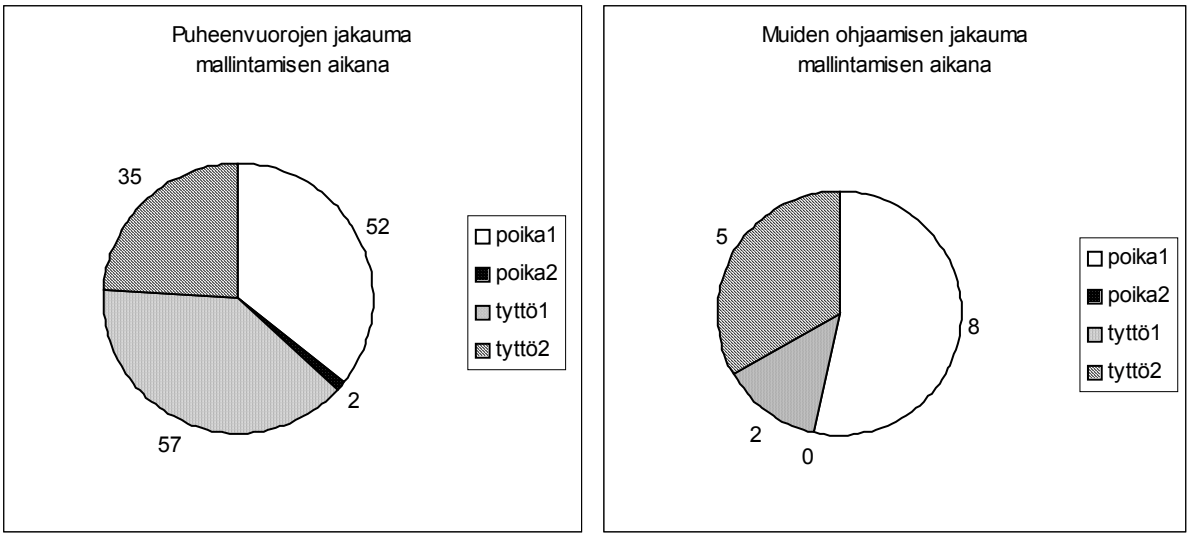
Kuva 23: Aloittelevien ryhmän mallintaminen.

Aloittelevien ryhmän mallintama liskon elinolosuhteiden tarkkailujärjestelmä (liite 10) oli rakenteeltaan yllättäen jopa monimutkaisempi kuin Kokeneempien ryhmän luoma ohjelma. Aloittelevien ryhmän ongelmanmäärittelyssä kohtaamat ongelmat ja sitä kautta vähemmälle jäänyt suunnittelu vaikutti lopputulokseen jopa positiivisesti. Ryhmä kohtasi ongelmia loogisten operaatioiden käytössä, mutta johdattelevien kysymysten ja pohdinnan avulla ongelmat saatiin ratkaistua. Ongelmana loogisten operaatioiden muodostamisessa Empirica Control -ohjelmistolla oli kahden IF-rakenteen peräkkäinen järjestys. Ryhmä yritti luoda operaation yhteen tai kahteen rinnakkaiseen IF-rakenteeseen. Jotta ryhmä kykeni ratkaisemaan ongel-

man, jouduimme johdattelemaan ryhmän toimintaa oikeaan suuntaan. Tämän jälkeen ryhmälle ei vastaavanlaisia virheitä enää syntynyt.

Kokeneempien ryhmän mallintamisprosessia ohjasi jo ongelmanmäärittelyvaiheessa pitkälle valmistunut suunnitelma. Ryhmän suunnittelema ohjelmanrakenne oli niin tarkka, että itse ohjelman kirjoittaminen onnistui suoraan paperilta mallia katsomalla. Ohjelmaan ei sisällytetty loogisia operaatioita tai muuttujia, joten ohjelmasta muodostui varsin suoraviivainen. Kehotuksistamme huolimatta ryhmä ei halunnut kehittää ohjelmaa sen enempää, sillä se palveli heidän tavoitteitaan sellaisenaan. Suurin osa mallintamiseen varatusta ajasta kului Kokeneempien ryhmältä testiajojen valmisteluun, koska ilmapallon asettaminen virkkauskoukulla kiinni massa-anturiin oli hankalaa. Tämä hankaluus johti siihen. että muut ryhmän jäsenet seurasivat hiljaa, kun poika 1 viritti laitteistoa.

Kokeneiden ryhmässä jäsenet neuvoivat toisiaan aktiivisesti, vaikka poika 2 osallistui koko mallintamisprosessin aikana ryhmän toimintaan vain satunnaisesti. Muut ryhmän jäsenet sen sijaan olivat tasavertaisesti ja jatkuvasti mukana ryhmän toiminnassa (kuva 24).



Kuva 24: Kokeneempien ryhmän mallintaminen.

Kokeneiden ryhmäläiset käyttivät mallintamisprosessissa selvästi mielikuvitusta. Varsinkin massa-anturin valjastaminen venytysvoimakkuutta mittaavaksi ratkaistiin mielenkiintoisesti. Ryhmän lopullisen ohjelman (liite 11) avulla tutkittiin valon läpäisevyyden muutosta ilmapalloa venytettäessä. Kokeneiden ryhmä hyödynsi selvästikin rajatun anturitarjonnan mahdollisimman tehokkaasti ja kekseliäästi. Empirica Control -ohjelmiston käyttö sujui ongel-

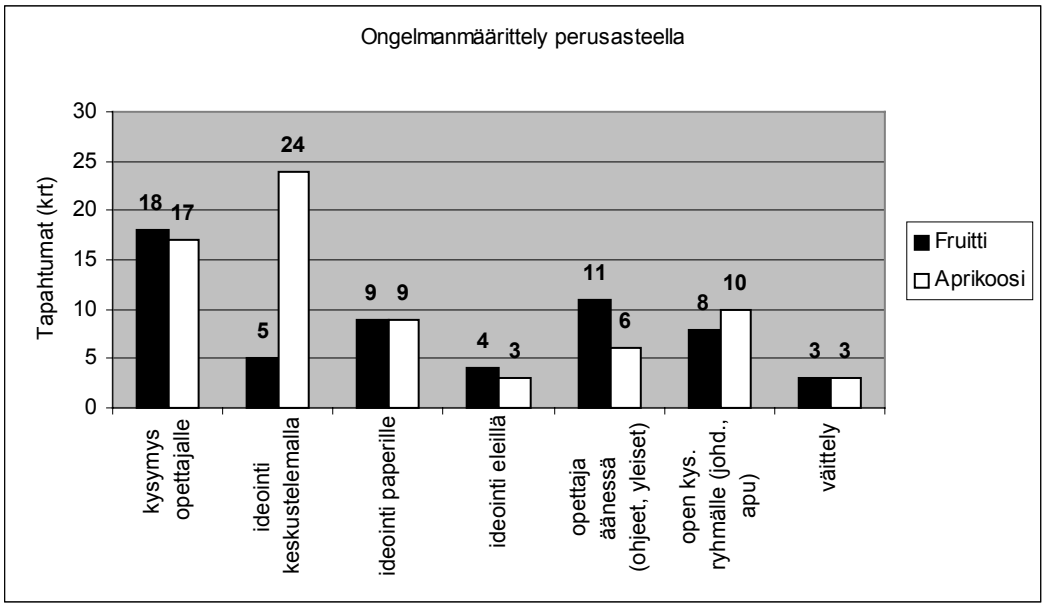
mitta, sillä ohjelma käytännössä kopioitiin paperilta ja ohjelma oli toimintakunnossa lähes välittömästi. Koko mallintamisprosessin aikana ryhmän toiminta oli todella itsenäistä, eikä ohjaustamme juurikaan tarvittu.

8 TULOKSET JA TULOSTEN ANALYSOINTI

Tässä luvussa tutustumme opetuskokeilussa kerätyn materiaalin avulla saavutettuihin johtopäätöksiin. Opetuskokeilun aikana kerätyt videomateriaalit analysoitiin ja tilastoitiin kvantitatiiviseen muotoon kutakin ryhmää, perusasteen Fruitti (liite 5) ja Aprikoosi (liite 4) sekä lukion Aloittelevia (liite 6) ja Kokeneempia (liite 7) varten. Käsittelemme opetuskokeiluun osallistuneiden ryhmien toimintaa, Empirica Control -ohjelmiston hyödyntämistä ohjelmoinnin oppimisessa ja käymme läpi Empirica-mittausjärjestelmän hyödyntämistä opetuksessa.

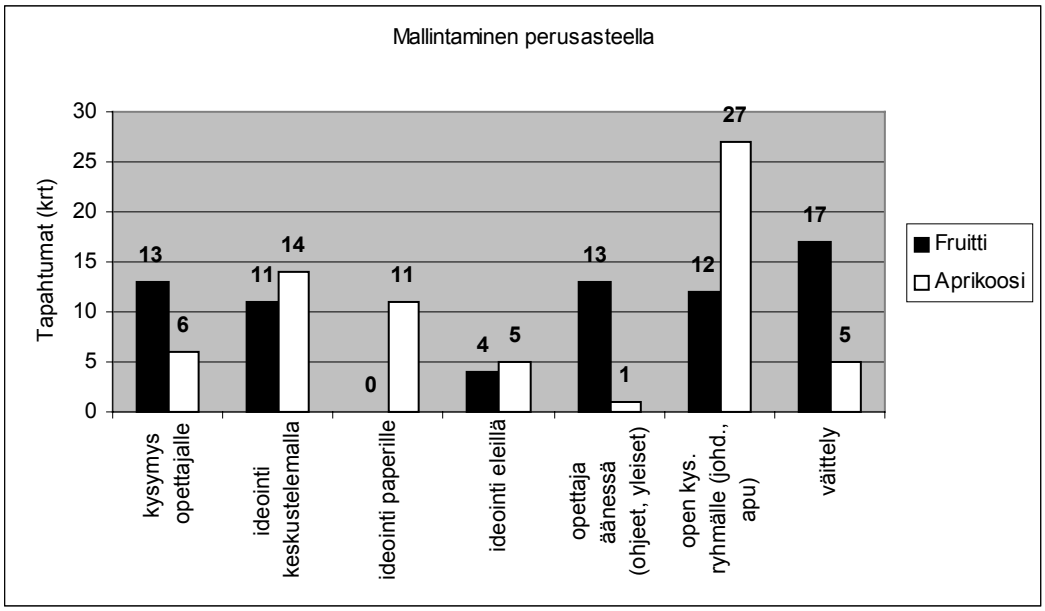
8.1 Ryhmien toiminta

Perusasteella ongelmanmäärittelyvaiheessa (kuva 25) erottui selvemmin kokeneemman Aprikoosi-ryhmän aktiivisuus ongelmien keksimisessä. Aprikoosi-ryhmä oli kokeneempi osittain alkuperäisen ryhmäjaon ja yhden tietotekniikkaa aiemmin harrastaneen Fruitti-ryhmän oppilaan sairastumisen takia. Vaikka Aprikoosi-ryhmä kyseli meiltä neuvoja lähes yhtä usein kuin kokemattomampi Fruitti-ryhmä, ongelmanmäärittelyssä ryhmän toiminta oli idearikkaampaa. Ryhmä keksi monia mahdollisia ongelmia, jotka voitaisiin ratkaista Empirica Controlin avulla. Ryhmä valitsi lopullisen ratkaistavan ongelman yhteistoimin. Meidän apuamme ryhmä ei ongelmien määrittelemisessä juurikaan tarvinnut. Heidän kysymyksensä olivat lähinnä yleisiä järjestelyjä sekä laitteistoa koskevia. Fruitti-ryhmälle meidän antamamme vastakysymykset olivat tärkeämpiä. Kysymyksen jälkeen ryhmä toimi hetken, jonka jälkeen meiltä haettiin uudelleen tukea työskentelyyn.



Kuva 25: Ongelmanmäärittely perusasteella.

Empirica-mittausjärjestelmän avulla suoritettu mallintaminen sujui molemmilta perusasteen ryhmiltä ilman suuria ongelmia ja odotettua nopeammin (kuva 26). Mallintamisvaiheessa Fruitti-ryhmä esitti edelleen meille kysymyksiä, joiden avulla ryhmäläiset hakivat tukea toimintaansa. Jouduimme osallistumaan ryhmän toimintaan huomattavasti enemmän kuin Aprikoosi-ryhmän toimintaan. Aprikoosi-ryhmä toimi itsenäisemmin ratkaisten yhteisesti rajattua ongelmaa. Tämä ryhmä käytti mallintamisvaiheessakin paperia ideoiden kirjaamiseen. Ideoiden avulla he pystyivät ratkaisemaan eteen tulleet ongelmat. Jouduimme esittämään Aprikoosi-ryhmälle kysymyksiä sen takia, että ryhmä ratkaisi määrittelemänsä ongelman todella nopeasti. Ryhmää pyrittiin kannustamaan laatimaansa ohjelman eteenpäin kehittämiseen, mutta ryhmä oli selkeästi tyytyväinen ensimmäiseen lopputulokseen. Mallintamisvaiheessa ryhmien sisällä käytiin huomattavasti enemmän väittelyitä kuin ongelmanmäärittelyn aikana. Fruitti-ryhmässä osasyy väittelyiden huomattavaan lisääntymiseen toisen päivän aikana voi olla ryhmän voimasuhteiden muuttuminen toisen pojan sairastuttua.

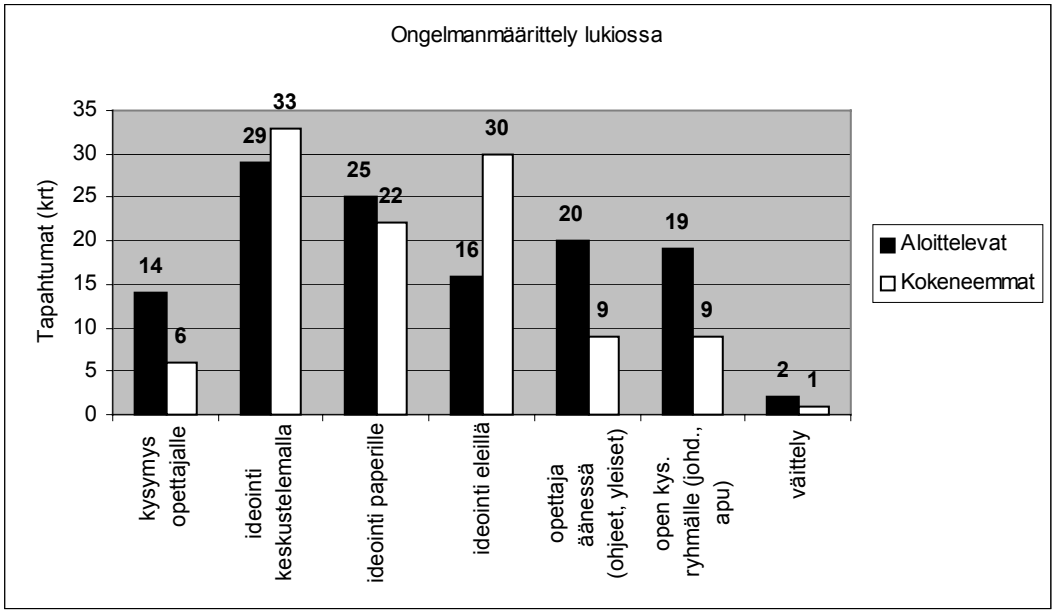


Kuva 26: Mallintaminen perusasteella.

Pennington et al. (1995) toteavat, että ohjelmoijien kokemus vaikuttaa heidän kykyynsä suunnitella ja määritellä ongelmia. Heidän mukaansa ohjelmoijan tietotaso vaikuttaa ongelman analysointiin ja osaongelmien laatimiseen. Samassa tutkimuksessa todetaan, että aloittelevat ohjelmoijat kuluttavat huomattavasti enemmän aikaa ongelman osakokonaisuuksien luomiseen ja hylkäämiseen. Perusasteella ryhmien toiminnassa oli nähtävissä samankaltaisia piirteitä. Aprikoosi-ryhmässä ohjelmointikokemusta oli selvästi Fruitti-ryhmää enemmän, mikä vaikutti osaltaan ongelman määrittelemiseen sekä mallintamiseen. Kokemattomampi Fruitti-ryhmä käytti huomattavasti enemmän opettajia tukena ongelmanmäärittelyn sekä mallintamisprosessien aikana.

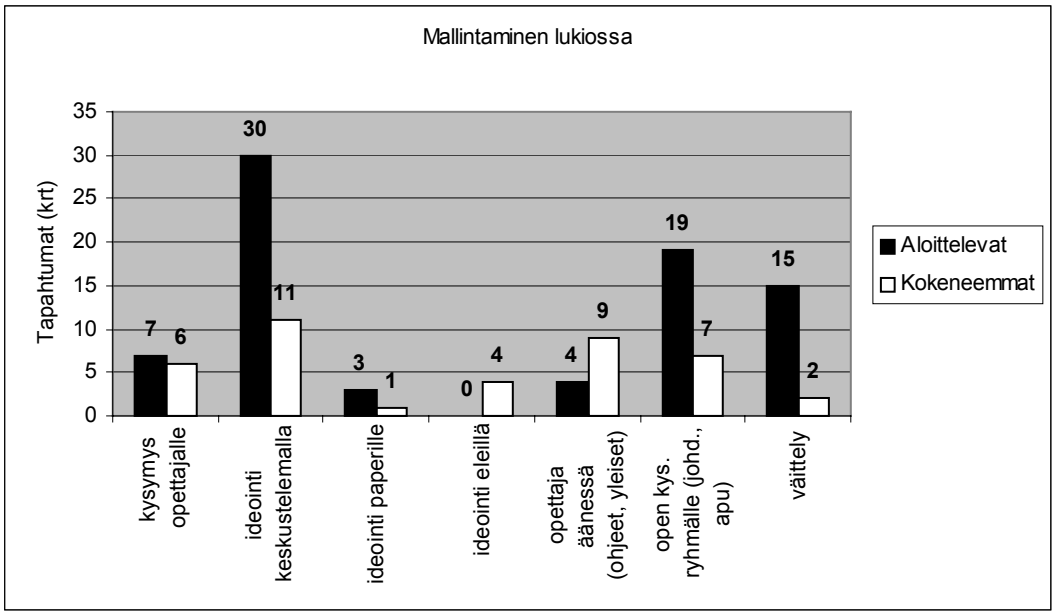
Lukiossa Kokeneempien ryhmä oli ongelmanmäärittelyssä vielä selvemmin aktiivisempi kuin perusasteen Aprikoosi-ryhmä. Kokeneempien ryhmä käytti huomattavasti vähemmän opettajaa apuna ongelmanmäärittelyssä keskittyen ryhmässä ideointiin. Aloittelevien ryhmässä ongelmanmäärittely tuotti suurempia ongelmia, joten meitä käytettiin apuna huomattavasti enemmän (kuva 27). Kokeneempien ohjelmoijien ryhmä toimi kuten Pennington et al. (1995) tutkimuksessaan havaitsivat. Ryhmäläisillä oli aiempaa tietoa ohjelmoinnista, minkä ansiosta ongelmanmäärittelemisessä ei kohdattu ongelmia. Aloittelevien ryhmän ohjelmointikokemuksen puute heijastui heidän ongelmanmäärittelyynsä. Ongelma-avaruutta ei hahmotettu läheskään yhtä hyvin kuin Kokeneempien ryhmässä. Molemmissa ryhmissä esiintyi ongelmanmää-

rittelyvaiheessa paljon ideointia keskustelemalla, mutta Aloittelevien ryhmässä keskustelun kimmokkeena käytettiin meidän apuamme. On myös huomattava, että Aloittelevien ryhmä ei käyttänyt paperia ideoidessaan ohjelmarakenteiden kuvaamista. Ryhmäläiset piirtelivät paperille muita asiaankuulumattomia kuvioita.



Kuva 27: Ongelmanmäärittely lukiossa.

Mallintamisvaiheessa Aloittelevien ryhmä käytti huomattavasti enemmän aikaa ideointiin kuin Kokeneempien ryhmä (kuva 28). Tämä johtuu ohjelmointikokemuksen puutteesta. Kokeneempien ryhmässä jo paperille laadittu ohjelman malli oli lähes täydellinen verrattuna ryhmän lopulliseen tuotokseen. Tästä johtuen ryhmässä ei juurikaan enää mallintamisvaiheessa väitelty eikä ideoitu. Ryhmä laati yksinkertaisen ohjelman joka toimii varmasti, kun taas Aloittelevien ryhmässä ohjelma laadittiin yritys-erehdys-periaatteen avulla. Aloittelevien ryhmässä ideointi jatkui koko mallintamisprosessin ajan ja ohjelmaa laadittaessa ryhmässä esiintyi usein väittelyjä. Aloittelevien ryhmä tarvitsi huomattavasti enemmän opettajien tukea mallintamisprosessissa.



Kuva 28: Mallintaminen lukiossa.

Niin lukiossa kuin perusasteella oli havaittavissa Penningtonin et al. (1995) havaitsemat erot kokeneiden ja aloittelevien ohjelmoijien välillä. Kokeneet ohjelmoijat eivät kohdanneet suurempia hankaluuksia ongelmanmäärittelyssä, kun taas aloitteleville ohjelmoijille ongelmanmäärittely tuotti suuria ongelmia, minkä vuoksi meiltä haettiin enemmän tukea.

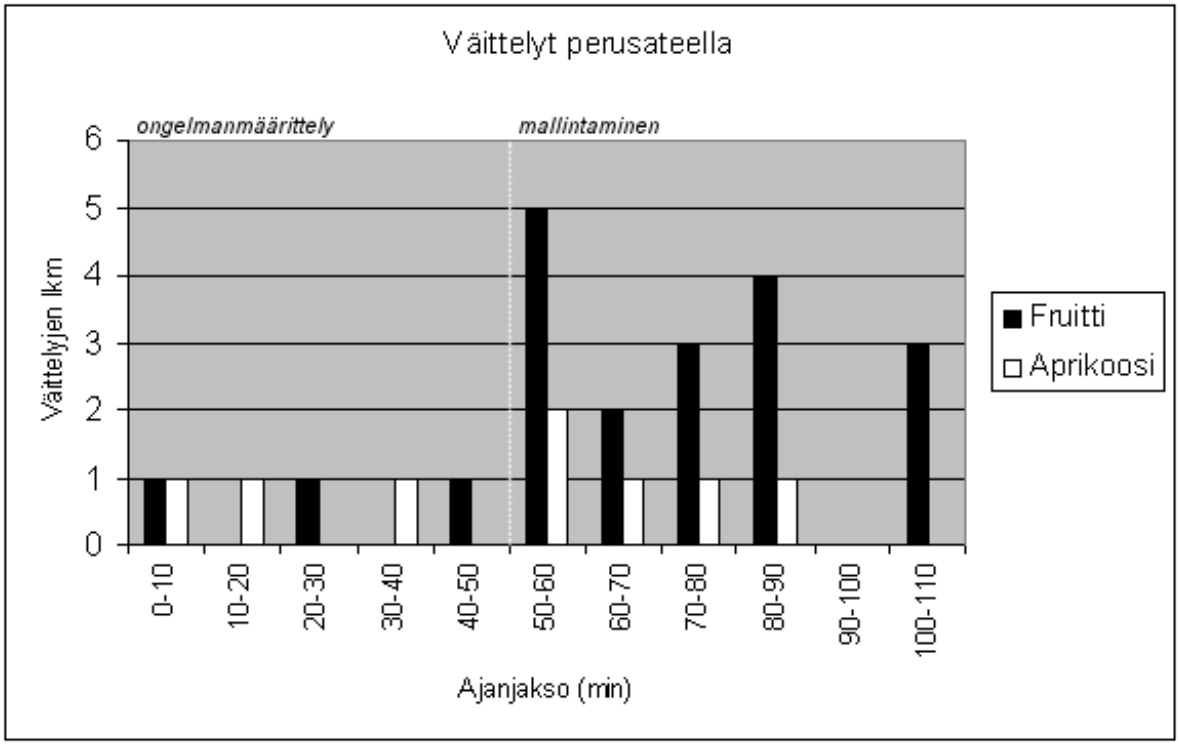
8.2 Empirica Control ja ohjelmoinnin oppiminen

Seuraavassa käsittelemme havaintoja ryhmien toimimisesta Empirica Control -ohjelmiston parissa. Lisäksi käsittelemme loppukyselyjen avulla kerättyä tietoa Empirica Control -ohjelmistosta sekä ohjelmoinnin perusrakenteiden oppimisesta.

8.2.1 Perusaste

Perusasteen oppilaat kritisoivat valmiiksi määriteltyjä ryhmiä ja ilmeisesti syystä. Heidän mukaansa ryhmätyöskentelyssä ei tavallisesti sallita omia ryhmiä, vaan opettaja suorittaa jaon. Ryhmien jakaminen ennalta ei välttämättä ollut kaikkein paras ratkaisu, vaikkakin uskomme ennalta määrättyjen ryhmien estäneen keskustelun harhautumisen aiheesta. Opetuskokeilun jälkeen heräsi myös kysymys, ovatko perusasteikäiset oppilaat vielä riittävän kypsiä toimimaan sekaryhmissä? Ainakin ryhmien toiminnan aikana ilmeni viitteitä kommunikointiongelmista poikien ja tyttöjen välillä (kuva 29). Toisaalta kolmen tunnin lähes yhtäjaksoinen

työskentely tietokoneen äärellä saattoi olla perusasteikäisille liikaa, vaikka välitunnit pidettiin normaalisti.



Kuva 29: Opetuskokeilun aikana tapahtuneet väittelyt perusasteella.

Empirica Control sai oppilailta hyvää palautetta lähinnä helppokäyttöisyydestään. Ohjelman kuvakkeet ja käyttöliittymä koettiin suurimmilta osin hyviksi. Negatiivista palautetta annettiin lähinnä ohjelman puutteellisista muokkaustoiminnoista. Kokonaisuudessaan opetuskokeilu koettiin mukavana vaihteluna normaaliin koulunkäyntiin. Ryhmien jäsenet eivät olleet aiemmin osallistuneet vastaavanlaisiin kokeiluihin.

Ohjelmoinnin perusrakenteiden oppimisessa havaittiin selvä jako aktiivisiin ja passiivisiin osallistujiin. Molempien ryhmien pojat vastasivat loppukyselyn (liite 13) ohjelmointirakenteita koskeviin kysymyksiin, kun taas kaikki tytöt eivät näihin kysymyksiin vastanneet. Fruitti-ryhmän ainoa poika vastasi kysymyksiin käyttäen Empirica Control -ohjelmiston kuvakkeita apunaan. Kuitenkin vastausten tarkkuudessa oli toivomisen varaa, vaikka käsitteet yhdistettiin selkeästi Empirica Control -ohjelmistoon. Aprikoosi-ryhmän pojat kykenivät suurempaan tarkkuuteen vastauksissaan. Aprikoosin poika 2 kuvasi silmukan seuraavasti: *”silmukka toistaa ohjelmaa niin monta kertaa kuin halutaan eikä ohjelmaa tarvitse kirjoittaa*

montaa kertaa uudestaan”. Aprikoosi-ryhmän tyttö 1 ja Fruitti-ryhmän tyttö 2 kykenivät vastaamaan annettuihin kysymyksiin kohtuullisen hyvin käyttäen apuna Empirica Control -ohjelmiston kuvakkeita. Kuitenkin vastausten tarkkuudessa oli toivomisen varaa.

Empirica Control -ohjelmisto sai perusasteella loppukyselyjen (liite 13) perusteella pelkästään hyvää palautetta. Suurin osa oppilaista piti ohjelmistoa helppokäyttöisenä ja selkeänä. Neljä oppilasta koki Empirica Control -ohjelmiston auttaneen heitä hahmottamaan ohjelmointia. Yhtä oppilasta lukuun ottamatta kaikki pitivät kokeiluun osallistumista hyödyllisenä, vaikkakin tytöt huomauttivat, etteivät he ymmärtäneet kaikkea opetettua. Ainoa oppilas, joka ei pitänyt opetuskokeilusta oli Fruitti-ryhmän tyttö 2, jonka osallistuminen ei ollut vapaaehtoista.

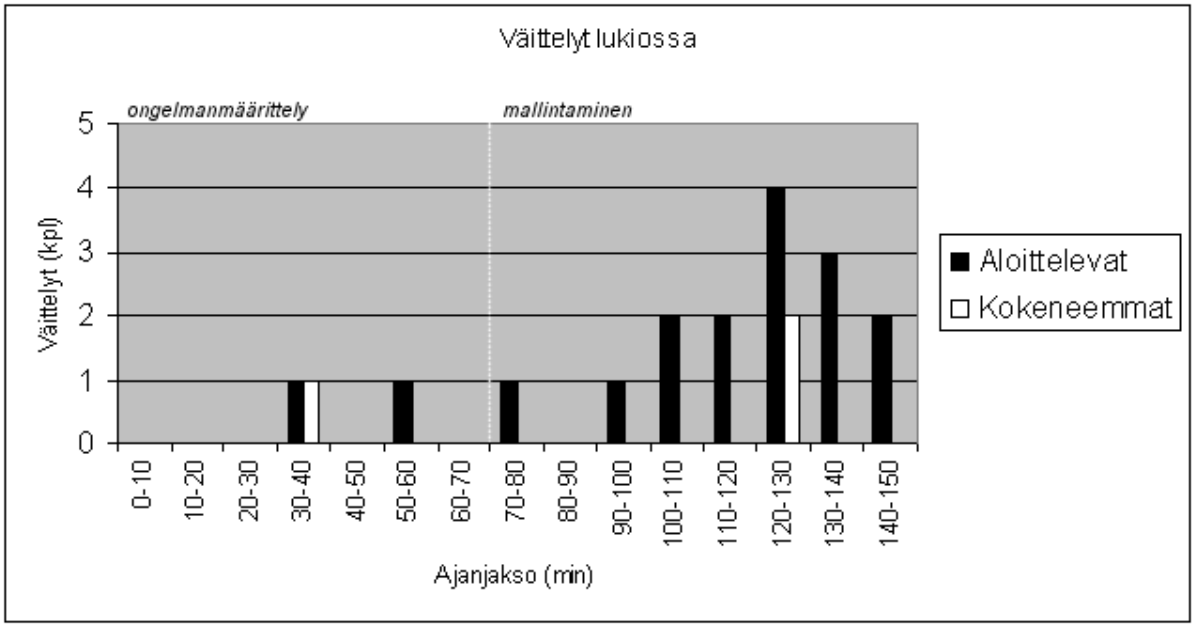
8.2.2 Lukio

Mallintamisprosessi sujui lukiolaisilla erittäin hyvin. Varsinkin Aloittelevien ryhmän suoritus yllätti positiivisesti. Keskustelu ryhmän jäsenten välillä oli vilkasta ja idearikasta, eikä mallintaminenkaan tuottanut ylitsepääsemättömiä ongelmia. Ryhmädynamiikka toimi erittäin hyvin molemmissa ryhmissä, vaikkakin Kokeneempien ryhmän toiminnassa poika 2 jäi muihin verrattuna selvästi taka-alalle. Loppukyselyssä hän itsekin mainitsi vähäisen osallistumisensa ongelmanmäärittelyyn ja mallintamiseen.

Suurimmaksi osaksi palaute Empirica Control -ohjelmistosta oli positiivista ja projektimuotoinen työskentely koettiin vaihteluna perinteiseen koulutyöhön. Empirica Control -ohjelmiston osalta kiitosta saivat selkeät kuvakkeet ja ohjelman muodostaminen visuaalisesti. Molemmat ryhmät kiinnittivät huomiota siihen, että ohjelmointi ei ole pelkkää rivien kirjoittamista. He ymmärsivät, että on olemassa myös erilaisia ohjelmointitapoja. Kokeneiden ryhmässä Empirica Control -ohjelmistolla ohjelmointi koettiin suhteellisen helpoksi, mutta silti opettavaiseksi ja mukavaksi toiminnaksi. Aloittelevien ryhmässä huomioitiin lisäksi ohjelman suoritusta kuvaavan sinisen pallon hyödyllisyys. Sinisen pallon avulla ohjelman suorituksen seuraaminen oli helppoa ja ongelmakohdat kyettiin paikantamaan varsin nopeasti.

Kaikista opetuskokeiluun osallistuneista lukion oppilaista kuitenkin yksi ilmoitti, että kokeilu innosti häntä ottamaan jatkossa valinnaisia ohjelmointikursseja. Useat oppilaat olivat päättäneet jo aiemmin valitsevatko jatkossa tietotekniikka- ja ohjelmointikursseja, eikä opetuskokeilu muuttanut heidän mielipiteitään.

Lukiossa ryhmien työskentelyssä esiintyi väittelyitä lähinnä mallintamisvaiheessa (kuva 30). Huomattavaa on se, että Kokeneiden ryhmässä väittelyitä oli vähän. Yhdessä laadittu yksityiskohtainen suunnitelma ei aiheuttanut ryhmässä erimielisyyksiä. Aloittelevien ryhmän kokemattomuus ohjelmoinnin parissa ilmeni useina väittelyinä, joiden avulla ryhmä haki yhteistä tapaa ratkaista ongelma Empirica Control -ohjelmiston avulla.



Kuva 30: Lukiolaisten väittelyt opetuskokeilun aikana.

Lukiolaisten vastaukset loppukyselyyn (liite 13) olivat ohjelmoinnin perusrakenteiden osalta kattavia. Kukaan lukion oppilaista ei käyttänyt selityksissään apuna Empirica Control -ohjelmiston kuvakkeita, vaan kaikki vastaukset olivat sanallisia. Suurin osa oppilaista kykeni selittämään silmukan ja ehtolauseen oikein. Vastauksissa hyödynnettiin esimerkkejä todellisesta maailmasta. Muuttujan suhteen lähes kaikilla oppilailla oli selviä hahmottamisongelmia. Tämä tosin ei yllättänyt, koska muuttujan käyttö jäi opetuskokeilun aikana vähäiseksi.

Lukiolaisista jokainen aiemmasta ohjelmointikokemuksesta riippumatta koki Empirica Control -ohjelmiston auttaneen ohjelmoinnin hahmottamisessa. Aloittelevien ryhmässä Empirica Control -ohjelmisto koettiin selkeänä ja helppona oppia, mutta ohjelman muokattavuudessa havaittiin selviä puutteita. Silmukan lisääminen jo olemassa olevan ohjelmarakenteen ympärille ei onnistu. Kokeneiden ryhmässä havaittiin samoja asioita kuin Aloittelevien ryhmässä, mutta Kokeneemmat tiedostivat myös Empirica Control -ohjelmiston rajalliset käyttömahdollisuudet. Kuitenkin Kokeneiden ryhmän poika 1 koki Empirica Control -ohjel-

miston liian helpoksi. Hänen mukaansa ohjelmistosta on hyötyä erityisesti aloitteleville ohjelmoijille.

Opetuskokeiluun suhtauduttiin lukiolaisten keskuudessa positiivisesti. Aloittelevien ryhmän tyttö 2 koki, että opetuskokeilu jopa muutti hänen käsitystään ohjelmoinnista. Opetettavat asiat ja Empirica Control -ohjelmistolla ohjelmoiminen koettiin lukiolaisten keskuudessa helpoiksi. Kokeneiden ryhmässä Empirica Control -ohjelmistolla ohjelmointi koettiin tavallista ohjelmointia mielekkäämmäksi ja nopeammaksi. Kuitenkin ryhmässä havaittiin Empirica Control -ohjelmiston rajoitetut käyttömahdollisuudet. Aloittelevien ryhmässä tyttö 2 ilmoitti Empirica Control -ohjelmiston huonoksi puoleksi myös antureiden rajatun määrän. Opetuskokeilussa käytössä ei ollut kuin kolme anturia ryhmää kohden. Avoimeksi jäi tarkoitettiinko vastauksessa antureiden kytkemismahdollisuutta Empirica Interfaceen vai erilaisten antureiden määrää.

8.3 Kuinka Empirica Control -ohjelmistoa voisi hyödyntää opetuksessa?

Opetuskokeilun aikana havaitsimme selkeitä etuja joita Empirica-mittausjärjestelmällä voidaan saavuttaa opetustilanteessa. Empirica Control -ohjelmisto toi selkeyttä ja konkreettisempaa näkökulmaa ohjelmointia kohtaan. Loppukyselyistä (liite 13) saaduista kommenteista havaittiin selkeästi, että ohjelmointi Empirica Control -ohjelmiston avulla oli hauskaa ja helppoa. Tämä kävi ilmi myös aiemmin ohjelmointia harrastaneiden vastauksista. Näiden havaintojen perusteella voidaan todeta, että Empirica-mittausjärjestelmä Empirica Control -ohjelmointiympäristöllä varustettuna palvelee opetustilanteessa varsinkin ohjelmointiin perehdyttävänä työkaluna. Empirica Control huolehtii ohjelman rakenteen muodostamisesta, mikä osaltaan vapauttaa käyttäjien voimavaroja luovuuden hyödyntämiseen. Lavosen et al. (2001a) mukaan ulkoa muistamisen minimointi olikin päämääränä Empirica Control -ohjelmistoa suunniteltaessa, jolloin käyttäjille jäisi enemmän mahdollisuuksia luovaan ongelmanratkaisuun.

Opetuskokeilun edetessä kävi selväksi, että Empirica Control -ohjelmiston käyttö koettiin mielekkääksi. Aiempaa ohjelmointikokemusta omaavat oppilaat suhtautuivat Empirica Control -ohjelmistoon myönteisesti ja kokivat sen avulla ohjelmoimisen hyödylliseksi ohjelmoinnin oppimisen kannalta. Kokemustemme perusteella Empirica Control -ohjelmisto voisi soveltua erittäin hyvin aloittelevan ohjelmoijan ensimmäiseksi ohjelmointikieleksi. Ohjelmoin-

nin perusrakenteisiin tutustuminen Empirica Control -ohjelmiston avulla madaltaa kynnystä ohjelmoinnin opiskeluun. Lisäksi Empirica Control -ohjelmiston visuaalinen ohjelmointiympäristö helpottaa yritys-erehdys-periaatteella tapahtuvaa oppimista. Opetuskokeilun aikana kerätystä materiaalista havaittiin, kuinka ryhmät uskalsivat kokeilla pelkäämättä virheitä ja erehtymisiä. Empirica Control -ohjelmiston avulla laaditut ohjelmat saadaan helposti suorituskelpoisiksi, sillä virheilmoitukset ovat selkeitä ja suomenkielisiä.

9 YHTEENVETO

Tutkimuksemme suurimmat ongelmat kohtasimme opetuskokeilun järjestelyissä. Aikataulujen sovittaminen koulujen opetussuunnitelmiin sopiviksi osoittautui hankalaksi. Kuitenkin Merilän perusasteen ja Lieksan lukion rehtoreiden avustuksella opetuskokeilut saatiin järjestettyä – tosin kuukausia alkuperäistä suunnitelmaa myöhemmin. Lisäksi opetuskokeiluun käytettävissä oleva aika oli rajallinen niin koulujen kuin tutkijoidenkin puolelta. Kokeiluun tarvittavien välineiden hankkiminen osoittautui oletettua hankalammaksi. Antureiden tilaaminen yliopistolle vei liikaa aikaa. Joensuun normaalikoulun avustuksella saimme käyttöömme opetuskokeilun suorittamiseen riittävän määrän antureita sekä toisen Empirica Interfacen.

Opetuskokeilussa saatujen aineistojen ja kokemusten perusteella voidaan todeta, että Empirica Controlin visuaalinen ohjelmointi helpottaa kynnystä ohjelmoinnin aloittamiseen. Tutkimukseen osallistuneista 15 oppilaasta yksi ilmoitti Empirica Controlin avulla suoritetun tutkimuksen innostaneen häntä tulevaisuudessa ohjelmoinnin opiskelun pariin. Samalla useat ilmoittivat tutkimuksen vahvistaneen heidän aiempaa päätöstään ottaa jatkossa ohjelmointiin liittyviä kursseja. Kuitenkin on todettava, että teimme tämänkaltaista tutkimusta ensimmäistä kertaa. Tämä on varmasti osaltaan vaikuttanut tutkimuksen suorittamiseen, aineiston analysointiin ja johtopäätösten kirjaamiseen.

Saavutettujen tulosten ja kokemusten perusteella voimme todeta, että ohjelmoinnin opetus voitaisiin hyvin aloittaa jo perusasteella. Opetuskokeilu perusasteella osoittautui sujuvan ennakkopeloista huolimatta yllättävän hyvin. Molemmat ryhmät kykenivät määrittelemään ratkaistavan ongelman hyödyntäen mielikuvitustaan. Lukiolla Aloittelevien ryhmälle ongelmanmäärittely tuotti huomattavasti enemmän ongelmia. Onko mahdollista, että erot ongelmanmäärittelemisessä lukion ja perusasteen välillä johtuvat pelkästään perusasteen oppilaiden mielikuvituksen vilkkaudesta.

Ohjelmoinnin perusrakenteiden opetuksessa Empirica Control osoittautui hyödylliseksi. Silmukan ja ehtolauseen opettaminen onnistui ilman suurempia hankaluuksia. Vaikka opetuskokeiluun osallistuneet oppilaat hyödynsivät silmukkaa ja ehtolausetta ohjelmissaan, loppukyselyssä käsitteiden selvittäminen tuotti useimmille hankaluuksia. Apuna selittämisessä käytettiin Empirica Controlin graafisia kuvakkeita. Muuttujien opetus Empirica Controlilla tuotti hankaluuksia. Tämä voi johtua siitä, että ohjelmoinnin perusrakenteiden opettamiseen

oli liian vähän aikaa. Siitä huolimatta muuttujien käyttäminen Empirica Control -ohjelmistolla on liian hankalaa.

Tutkimuksen aikana Empirica Control -ohjelmiston käytössä havaittiin ongelmia lähinnä loogisten operaatioiden käytössä. Vaikka oppilaat olisivat ymmärtäneet loogisen operaation merkityksen, sen varsinainen lisääminen ohjelmaan hiiren avulla tuotti hankaluuksia. Hiiren painallukset eksyivät vääriin kohtiin. Empirica-mittausjärjestelmän toiminnassa havaittiin lukiossa ongelmia. Massa-anturi mittasi negatiivisia arvoja vaikka anturi oli taarattu oikein.

Kerätyn aineiston analysoinnissa ongelmia tuotti aiemman kokemuksen puute. Videoiden analysoinnissa päädyttiin muuntamaan aineisto kvantitatiiviseen muotoon, jolloin vertailu eri ryhmien välillä helpottui. Ryhmien toiminnassa havaittiin eroja jäsenten ohjelmointikokemuksen mukaisesti. Vaikka perusasteella pyrkimyksenä oli muodostaa samantasoiset ryhmät, yhden oppilaan sairastuminen ennen mallintamisvaihetta vaikutti ryhmän kokonaisuuteen. Tämän seurauksena molemmissa opetuskokeiluissa ryhmien välille muodostui tasoero ohjelmointikokemuksen mukaan, mikä ilmeni myös ryhmien toiminnassa.

Tutkimuksemme aikana heräsi kysymyksiä, jotka voisivat olla aiheena mahdollisille jatkotutkimuksille. Empirica Control -ohjelmiston käytettävyysarvioinnin (esimerkiksi kognitiiviset ulottuvuudet) avulla voitaisiin tutkia, voidaanko käyttöliittymää kehittää palvelemaan opetusta paremmin. Pohdimme myös, voidaanko ohjelmoinnin opettamista kehittää Empirica Controlin avulla. Tämä voisi tapahtua esimerkiksi tutustuttamalla oppilaat ohjelmointiin Empirica Controlin avulla ennen perinteiselle ohjelmointikurssille osallistumista. Opetuskokeilun aikana heräsi myös kysymys, voisiko Empirica Control -ohjelmiston käyttö perusasteella integroituna esimerkiksi luonnontieteiden opettamiseen aktivoida oppilaita kiinnostumaan ohjelmoinnista. Ryhmien toiminnasta perusasteella voisi myös suorittaa tarkempia tutkimuksia. Saataisiinko tytöt innostumaan ohjelmoinnista enemmän, jos he toimisivat omissa ryhmissään. Tässä tilanteessa he saisivat ratkaista itseään kiinnostavia ongelmia esimerkiksi Empirica Controlin avulla.

LÄHTEET

Anderson, J., Naps, T. 2001. A Context for the Assesment of Algorithm Visualization Systems as Pedagogigal Tools. *Proceedings of the First Program Visualization Workshop* (Ed. Sutinen, E.). University of Joensuu, Joensuu, 121-130.

Ben-Ari, M. 2001. *Constructivism in Computer Science Education.* Journal of Computers in Mathematics & Science Teaching, **20**(1):45-73.

Bogdan, R., Biklen, S. 1998. *Qualitative Research for Education – An Introduction to Theory and Methods*. Allyn & Bacon, Boston.

Bruner, J. S. 1962. *On knowing: Essays for the left hand*. Cambridge, MA: Harvard University Press, Cambridge.

Card, S., Mackinlay, J., Schneiderman, B. 1999. *Readings in Information Visualization: Using Vision to Think*. Morgan Kauffmann Publishers Inc., San Francisco.

Crapo, A. W., Waisel, L. B., Wallace, W. A., Willemain, T. R. 2000. Visualization and The Process of Modeling: A Cognitive-theoretic View. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, Boston, 218-226.

Denscombe, M. 1998. *The Good Research Guide for Small-Scale Social Research Projects*. Biddles, UK.

Derry, S., Hawkes, L. 1993. *Local Cognitive Modelling of Problem-Solving Behavior: An Application of Fuzzy Theory. Computers as Cognitive Tools* (Ed. Lajoie, S., Derry, S.), Lawrence Erlbaum Associates, Publishers, 107-140.

Enkenberg J. 1998. Uutta pedagogiikkaa etsimässä. *Opetus, oppiminen ja vuorovaikutus*. (Ed. Julkunen, M-L.), WSOY, Porvoo, 158-176.

Erätuuli, M., Leino, J., Yli-Luoma, P. 1994. *Kvantitatiiviset analyysimenetelmät ihmistieteissä*. West Point Oy, Rauma.

Green, T., Blackwell, A. 1998. *Cognitive Dimensions of Information Artefacts: a tutorial.* WWW-sivusto, http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDtutorial.pdf (3.5.2002).

Hassinen, K. 1986. Tietotekniikan perusopetuksen ohjelmointikielistä. *Matemaattis-luonnontieteellisen tiedekunnan raporttisarja* (Ed. Erkama, T., Pasanen, S.). Joensuun yliopisto, Joensuu.

Järvinen, E-M. 1998. The Lego/Logo Learning Environment in Technology Education: An Experiment in a Finnish Context. *Journal of Technology Education* (Ed. LaPorte, J.). **9**(2):47-59.

Kamada, T., Kawai, S. 1991. A General Framework for Visualizing Abstract Objects and Relations. *ACM Transactions on Graphics*. **10**(1):1-39.

Khuri, S. 2001. Designing Effective Algorithm Visualizations. *Proceedings of the First Program Visualization Workshop* (Ed. Sutinen, E.). University of Joensuu, Joensuu, 1-12.

Kocijancic, S. 2001. Mechatronics as a Challenge for Teaching Technology in Secondary Education. *Proceedings of the 31st Annual Conference in Frontiers in Education '01*. **1**, T2E-1-T2E-4.

Kohlberg, E., Orlev, N. 2001. Robotics Learning as a Tool for Integrating Science-Technology Curriculum in K-12 Schools. *Proceedings of the 31st Annual Conference in Frontiers in Education '01*. **1**, T2E-12-T2E-13.

Lavonen, J., Kåll, L., Simbierowichz, P., Knuuttila, M., Meisalo, V. 1996. *Empirica Control*. Printel Oy, Vantaa.

Lavonen, J.M, Meisalo, V.P., Lattu, M. 2001a. Problem Solving with an Icon oriented Programming Tool: A Case Study in Technology Education. *Journal of Technology Education* (Ed. LaPorte, J.). **12**(2):21-34.

Lavonen, J.M., Meisalo, V.P., Lattu, M. 2001b. Visual Programming: Basic Structures Made Easy. *Proceedings of the First Program Visualization Workshop* (Ed. Sutinen, E.). University of Joensuu, Joensuu, 163-177.

Levy, R., Ben-Ari, M., Uronen, P. 2000. The Jeliot 2000 Program Animation System. *Journal of Visual Languages and Computing*. (Accepted for publication).

Levy, R., Ben-Ari, M., Uronen, P. 2001. An Extended Experiment with Jeliot. *Proceedings of the First Program Visualization Workshop* (Ed. Sutinen, E.). University of Joensuu, Joensuu. 131-140.

Lund, H. 1999. AI in Children's Play with LEGO Robots. *Proceedings of AAAI 1999 Spring Symposium Series* (Ed. Dobson, D., Forbus, K.), AAAI Press, California, 60-63.

Mehtäläinen, J. 1992. *Tiedollinen kasvatus ja ajattelun kehittäminen*. Opetushallitus, valtion painatuskeskus, Helsinki

Meisalo, V., Sutinen, E., Tarhio, J. 2000. *Modernit oppimisympäristöt.* Tietosanoma, Juva.

Meisalo, V., Sutinen, E., Tarhio, J. 1997. CLAP: Teaching Data Structures in a Creative Way. *Proceeding of the ITiCSE '97, Integrating Teaching into Computer Science Education* (Ed. Davies, G.), Uppsala University, Uppsala, 117-119.

Miglino, O., Lund, H., Cardaci, M. 1999. Robotics as an Educational Tool. *Journal of Interactive Learning Research*. **10**(1):25-48.

Miraftabi, R. 2001. Intelligent Agents in Program Visualizations: A Case Study With Seal. *Proceedings of the First Program Visualization Workshop* (Ed. Sutinen, E.). University of Joensuu, Joensuu, 53-58.

Myers, B. 1986. Visual Programming, Programming by Example and Program Visualization: A Taxonomy. *Proceedings of Human Factors in Computing Systems* (Ed. Mantei, M., Orbeton, P.). ACM Press, New York, 59-66.

Papert, S. 1980. *Mindstorms, Children, Computers and Powerful Ideas*. Basic Books, New York.

Pennington, N., Lee, A.Y., Rehder, B. 1995. Cognitive Activities and Levels of Abstraction in Procedural and Object-Oriented Design. *Human-Computer Interaction*. Hillsdale, New Jersey **10**(2&3):121-128.

Petre, M., Blackwell, A., Green, T. 1998. Cognitive Questions in Software Visualization. *Software Visualization* (Ed. Stasko, J., Domingue, M., Brown, M., Price, B.). MIT Press, Massachusetts, 453-480.

Rauste-von Wright, M-L. 1997. *Opettaja tienhaarassa: konstruktivismia käytännössä*. Atena, Jyväskylä.

Roman, G.-C., Cox, K. C. 1992. Program Visualization: The Art of Mapping Programs to Pictures. *Proceedings of the 14th International Conference on Software Engineering*. ACM Press, Melbourne. 412-420.

Saariluoma, P. 2001. Psychological Problems in Program Visualization. *Proceedings of the First Program Visualization Workshop* (Ed. Sutinen, E.). University of Joensuu, Joensuu, 13-27.

Sahlberg, P., Leppilampi, A. 1994. *Yksinään vai yhteisvoimin? – yhdessäoppimisen mahdollisuuksia etsimässä*. Helsingin yliopisto, Yliopistopaino, Helsinki.

Sutinen, E., Tarhio, J. 2001. Teaching to Identify Problems in a Creative Way. *Proceedings of the 31st Annual Conference in Frontiers in Education '01*. **1**, T1D-8-T1D-13.

Syrjälä, L., Ahonen, S., Syrjäläinen, E., Saari, S. 1994. *Laadullisen tutkimuksen työtapoja*. West Point Oy, Rauma.

Teasley, S., Roschelle, J. 1993. Constructing a Joint Problem Space: The Computer as a Tool for Sharing Knowledge. Computers as Cognitive Tools (Ed. Lajoie, S., Derry, S.), Lawrence Erlbaum Associates, New Jersey, 229-258.

Vaherva, T., Ekola, J. 1986. *Aikuisten opettamisen taito*. Yleisradio, Helsinki.

Waisel, L. B., Wallace, W. A., Willemain, T. R. 1999. Visualizing Modeling Heuristics: An Exploratory Study. Proceedings of the 1999 International Conference on Information Systems. Charlotte, North Carolina, 166-177.

Wang, E., Wang, R. 2001. Using Legos and Robolab (Labview) with Elementary School Children. *Proceedings of the 31st Annual Conference in Frontiers in Education '01*. **1**, T2E-11.

LIITTEET

Liite 1: Merilän perusasteen tuntisuunnitelma

Keskiviikko 20.2.2002

Tunti 1:
Tavoitteena selvittää oppilaille tietokoneen olemus – kaikki toiminnot neuvottava tietokoneelle. Lisäksi oppilaiden tulisi ymmärtää mitä ohjelmointi on sekä ymmärtää ohjelmoinnin perusrakenteiden tarkoitus ja toiminta.

- Mikä tutkimuksen tarkoitus on? Mitä tutkitaan, miksi ja miten?
- Mikä tietokone on?
- Keskustelua, miten tietokone toimii?
- Mitä on ohjelmointi? Silmukka? Muuttuja? Ehto? (käydään läpi Empirica Controlin ikoneiden avulla)
- Liikuntasalissa tai luokassa ohjelmoidaan toisiamme toimimaan

Tunti 2:
Tavoitteena ymmärtää mikä Empirica-mittausjärjestelmä on ja varsinkin Empirica Control -ohjelmiston käyttämiseen tarvittavien tietojen saavuttaminen.

- Tarvittaessa jatketaan 1. tunnin aiheita
- Empirica Controlin esittely
- Empirica Controlin anturit
- Helppo esimerkkiohjelma – pienet tehtävät joka ryhmälle

Tunti 3:
Tavoitteena ryhmissä määritellä Empirica Controlin avulla toteutettavissa oleva ongelma, sekä laatia suunnitelma toteutuksesta.

- Ohjelmointiharjoituksia Empirica Controlilla – yksinkertaisista vaikeampiin
- Lämpötilan mittaaminen jatkuvasti
    - Äänimerkki, kun saavutetaan tietty piste
    - Ilmoitus kun lämpötila yli 23^o tai alle 19^o jne.
    - Harjoitellaan samalla lokikirjan käyttöä
- Ongelman määrittely. Millainen käytännön ongelma voitaisiin ratkaista Empirica Controlin avulla?
- Ryhmät keräävät ideoita paperille joista keskustellaan koko joukon kesken. Ongelmia? Haasteita? Muuta?
- Valitaan ryhmän keskuudessa aihe, johon ”ope” antanut hyväksynnän ja joka voidaan toteuttaa Empirica Controlin ja antureiden avulla
- Laaditaan suunnitelma paperille ja alustava ohjelman kuvaus paperille

Tunti 4:

- Ongelman määrittely

Keskiviikko 27.2.2002

Tunti 5:
Tavoitteena palauttaa Empirica Controlin käyttö mieleen sekä ratkaista määritelty ongelma sen avulla.

- Kertaustehtävien läpikäynti yhdessä
- Ongelman määrittely
- Mallinnetaan ja ohjelmoidaan

Tunti 6:
- Mallinnetaan ja ohjelmoidaan

Tunti 7:
- Mallinnetaan ja ohjelmoidaan

Tunti 8:
Tunnin tarkoituksena tutustua toisen ryhmän laatimaan ohjelmaan sekä tiedustella oppilaiden kokemuksia Empirica Control -ohjelmistosta.
- Esitellään työt
- Palautekeskustelu
- Loppukyselyn täyttäminen
- Projektin päättäminen: Muuttuiko suunniteltu ohjelma rakenteellisesti paljon lopulliseksi ohjelmaksi? Miksi? Mitä oli ajateltu ja kuviteltu väärin? Muuta?.
- Tulosten esittely muille ryhmille, keskustelu Empirica Controlista ja projektista.
- Palautekyselyn täyttäminen.

Liite 2: Lieksan lukion tuntisuunnitelma

Keskiviikko 30.1.2002

Tunti 1:
Tavoitteena selvittää oppilaille mitä ohjelmointi on sekä ohjelmoinnin perusrakenteiden tarkoitus ja toiminta.

- Selvittää kokeen tarkoitus: mitä tutkitaan, miksi ja miten tutkitaan
- Mikä Empirica on?
- Mitä on ohjelmointi?
- Selvitetään käsitteet ehto, silmukka, muuttuja ja proseduraalinen rakenne yhdistämällä ne Empirica Controlin ikoneihin ja ohjelman rakentamiseen

Tunti 2:
Tavoitteena ymmärtää mikä Empirica-mittausjärjestelmä on ja varsinkin Empirica Control -ohjelmiston käyttämiseen tarvittavien tietojen saavuttaminen.

- Empirica Control -ohjelmointi – helppo esimerkki, jossa mukana ehto, silmukka, muuttuja, and tai or -ehto
- Lokikirjan käyttö
- Ohjelman ajaminen ja ohjelman kulun seuraaminen osoittimen avulla
- Ongelman määrittely - Pohtikaa käytännön ongelmaa, jonka voisitte ratkaista ja mallintaa käytettävissä olevien välineiden (Empirica Control & anturit) avulla

Tunti 3:
Tavoitteena ryhmissä määritellä Empirica Controlin avulla toteutettavissa oleva ongelma, sekä laatia suunnitelma toteutuksesta.

- Ongelman määrittely ja raportointi, paperille mallintaminen
- Alustava, Empirica Control -ikoneilla ja/tai sanallinen suunnitelma ohjelman toiminnasta

Torstai 31.1.2002.

Tunti 4:
Tavoitteena palauttaa Empirica Controlin käyttö mieleen sekä ratkaista määritelty ongelma sen avulla.

- Ohjelmointi ja mallintaminen

Tunti 5,6:
Tunnin tarkoituksena tutustua toisen ryhmän laatimaan ohjelmaan sekä tiedustella oppilaiden kokemuksia Empirica Control -ohjelmistosta.

- Ohjelmointi ja mallintaminen
- Projektin päättäminen: Muuttuiko suunniteltu ohjelma rakenteellisesti paljon lopulliseksi ohjelmaksi? Miksi? Mitä oli ajateltu ja kuviteltu väärin? jne.
- Tulosten esittely muille ryhmille, keskustelu Empirica Controlista ja projektista.
- Palautekyselyn täyttäminen.

Liite 3: Kotitehtävät

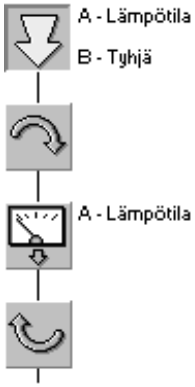
Kertaustehtäviä Empirica Control -ohjelmoinnista

Seuraavassa on muutamia esimerkkejä ohjelmista. Kerro omin sanoin mitä ohjelma mielestäsi tekee. Kuvaa ohjelman kulku, ja jos mahdollista kerro kuinka kauan ohjelmaa suoritetaan (tai monta kertaa).

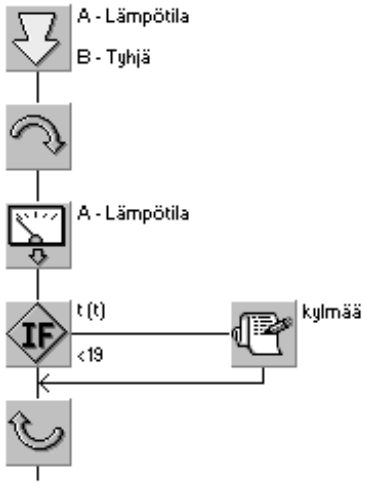
Ohjelmat käydään läpi ensi keskiviikkona yhdessä, mutta yritä parhaasi jo kotona! Jos mahdollista, tee tehtävät maanantai tai tiistai-iltana, jolloin voit muistella oppimaasi ennen keskiviikon tunteja.

Harri & Kai

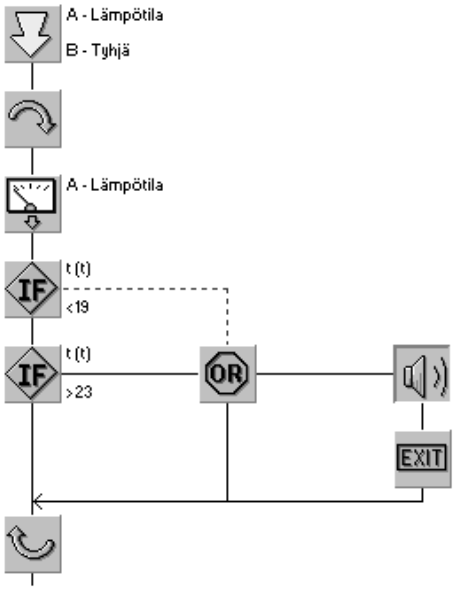
Ohjelma 1.



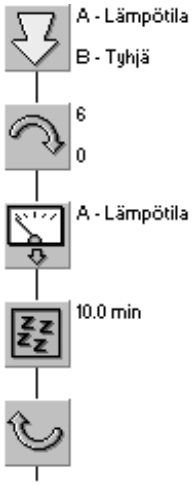
Ohjelma 2.



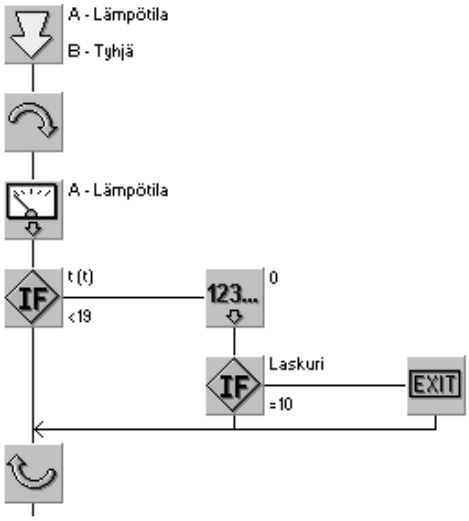
Ohjelma 3.



Ohjelma 4.



Ohjelma 5.



Liite 4: Perusaste, Aprikoosi-ryhmän videoanalyysi

	50-60	40-50	30-40	20-30	10-20	0-10	Mallintaminen alkaa:	2. pvä 0-10	30-40	20-30	10-20	Ongelm. määr. 0-10	**Perusaste - Aprikoosi** min
6	1	2	1	0	2	0	17	1	3	2	4	7	kysymys opettajalle
28	5	2	1	4	10	6	10	3	4	1	1	1	kysym., idean johdattelua ryhmässä
10	4	2	2	1	1	0	7	0	0	4	2	1	keskustelu open kanssa
14	3	1	3	3	2	2	24	0	4	6	7	7	ideointi keskustelemalla
11	2	2	2	0	2	3	9	2	2	0	0	5	ideointi paperille
5	1	1	1	0	2	0	3	0	0	0	0	3	ideointi eleillä
1	0	1	0	0	0	0	6	0	2	0	2	2	opettaja äänessä (ohjeet, yleiset)
27	5	6	6	5	4	1	10	0	0	5	1	4	open kys. ryhmälle (johd., apu)
5	0	0	1	1	1	2	3	0	1	0	1	1	väittely
9	4	1	2	0	2	0	4	1	1	1	1	0	hiljaisuus
10	0	2	3	1	0	4	5	2	1	0	0	2	naurua
2	0	0	0	1	0	1	5	1	0	1	2	1	välineiden tutkiminen
3	0	1	1	0	0	1	3	1	1	0	0	1	häiriöinti
0	0	0	0	0	0	0	0	0	0	0	0	0	ongelmia ryhmässä
													ÄÄNESSÄ:
52	9	10	7	10	10	6	44	5	9	9	12	9	Poika 1
41	7	7	5	7	9	6	36	1	6	10	10	9	Poika 2
21	0	4	2	4	5	6	19	4	4	4	4	3	Tyttö 1
11	1	3	1	3	1	2	7	3	0	2	1	1	Tyttö 2
													NEUVOO MUITA:
15	0	1	2	2	5	5	7	0	5	2	0	0	Poika 1
12	0	1	1	3	3	4	2	0	0	0	2	0	Poika 2
0	0	0	0	0	0	0	0	0	0	0	0	0	Tyttö 1
0	0	0	0	0	0	0	0	0	0	0	0	0	Tyttö 2
													AKTIVOI MUITA:
4	0	0	0	1	1	2	4	1	3	0	0	0	Poika 1
0	0	0	0	0	0	0	0	0	0	0	0	0	Poika 2
0	0	0	0	0	0	0	0	0	0	0	0	0	Tyttö 1
0	0	0	0	0	0	0	0	0	0	0	0	0	Tyttö 2
													POISTUU RYHMÄSTÄ:
2	0	0	0	0	0	2	3	2	1	0	0	0	Poika 1
0	0	0	0	0	0	0	14	3	4	6	1	0	Poika 2
0	0	0	0	0	0	0	3	0	1	1	1	0	Tyttö 1
1	0	0	1	0	0	0	5	1	1	3	0	0	Tyttö 2
													SEURAA TOISTA RYHMÄÄ:
1	0	1	0	0	0	0	0	0	0	0	0	0	Poika 1
2	0	1	0	0	0	1	1	0	1	0	0	0	Poika 2
5	0	1	3	1	0	0	3	1	0	0	0	2	Tyttö 1
9	0	1	5	3	0	0	2	1	0	0	1	0	Tyttö 2

Liite 5: Perusaste, Fruitti-ryhmän videoanalyysi

Perusaste - Fruitti								Mallintaminen alkaa:	2. pvä				Ongelm. määr.
min		50-60	40-50	30-40	20-30	10-20	0-10		0-10	30-40	20-30	10-20	0-10
kysymys opettajalle	13	2	1	1	3	5	1	18	8	1	2	4	3
kysym., idean johdattelua ryhmässä	6	2	0	1	1	0	2	5	0	0	4	1	0
keskustelu open kanssa	8	1	0	4	2	1	0	9	4	2	1	1	1
ideointi keskustelemalla	11	4	0	3	1	3	0	5	0	0	1	0	4
ideointi paperille	0	0	0	0	0	0	0	9	3	0	0	3	3
ideointi eleillä	4	1	0	1	1	0	1	4	0	0	3	0	1
opettaja äänessä (ohjeet, yleiset)	13	1	3	2	2	2	3	11	1	2	2	1	5
open kys. ryhmälle (johd., apu)	12	7	1	0	4	0	0	8	2	1	3	1	1
väittely	17	3	0	4	3	2	5	3	1	0	1	0	1
hiljaisuus	9	1	2	1	2	3	0	14	1	5	2	4	2
naurua	7	0	2	4	1	0	0	4	0	1	0	0	3
välineiden tutkiminen	6	1	0	0	2	2	1	4	2	0	2	0	0
häiriöinti	0	0	0	0	0	0	0	1	0	0	0	0	1
ongelmia ryhmässä	2	0	0	1	1	0	0	0	0	0	0	0	0
ÄÄNESSÄ:													
Poika 1	56	11	6	10	9	12	8	36	7	4	10	7	8
Poika 2	0							4		0	2	0	2
Tyttö 1	41	4	4	8	7	9	9	24	5	3	6	4	6
Tyttö 2	40	5	5	6	7	7	10	18	4	4	3	2	5
NEUVOO MUITA:													
Poika 1	19	2	2	2	2	5	6	4	0	2	2	0	0
Poika 2	0							1		0	1	0	0
Tyttö 1	0	0	0	0	0	0	0	0	0	0	0	0	0
Tyttö 2	8	0	0	0	1	3	4	0	0	0	0	0	0
AKTIVOI MUITA:													
Poika 1	5	1	0	0	0	1	3	6	2	0	3	0	1
Poika 2	0							0		0	0	0	0
Tyttö 1	0	0	0	0	0	0	0	0	0	0	0	0	0
Tyttö 2	0	0	0	0	0	0	0	0	0	0	0	0	0
POISTUU RYHMÄSTÄ:													
Poika 1	1	0	0	1	0	0	0	4	0	1	0	2	1
Poika 2	0							1		1	0	0	0
Tyttö 1	1	0	1	0	0	0	0	3	2	1	0	0	0
Tyttö 2	0	0	0	0	0	0	0	4	3	1	0	0	0
SEURAA TOISTA RYHMÄÄ:													
Poika 1	0	0	0	0	0	0	0	4	1	0	0	0	3
Poika 2	0							8		1	0	4	3
Tyttö 1	1	0	0	1	0	0	0	2	1	0	0	0	1
Tyttö 2	0	0	0	0	0	0	0	3	1	1	0	0	1

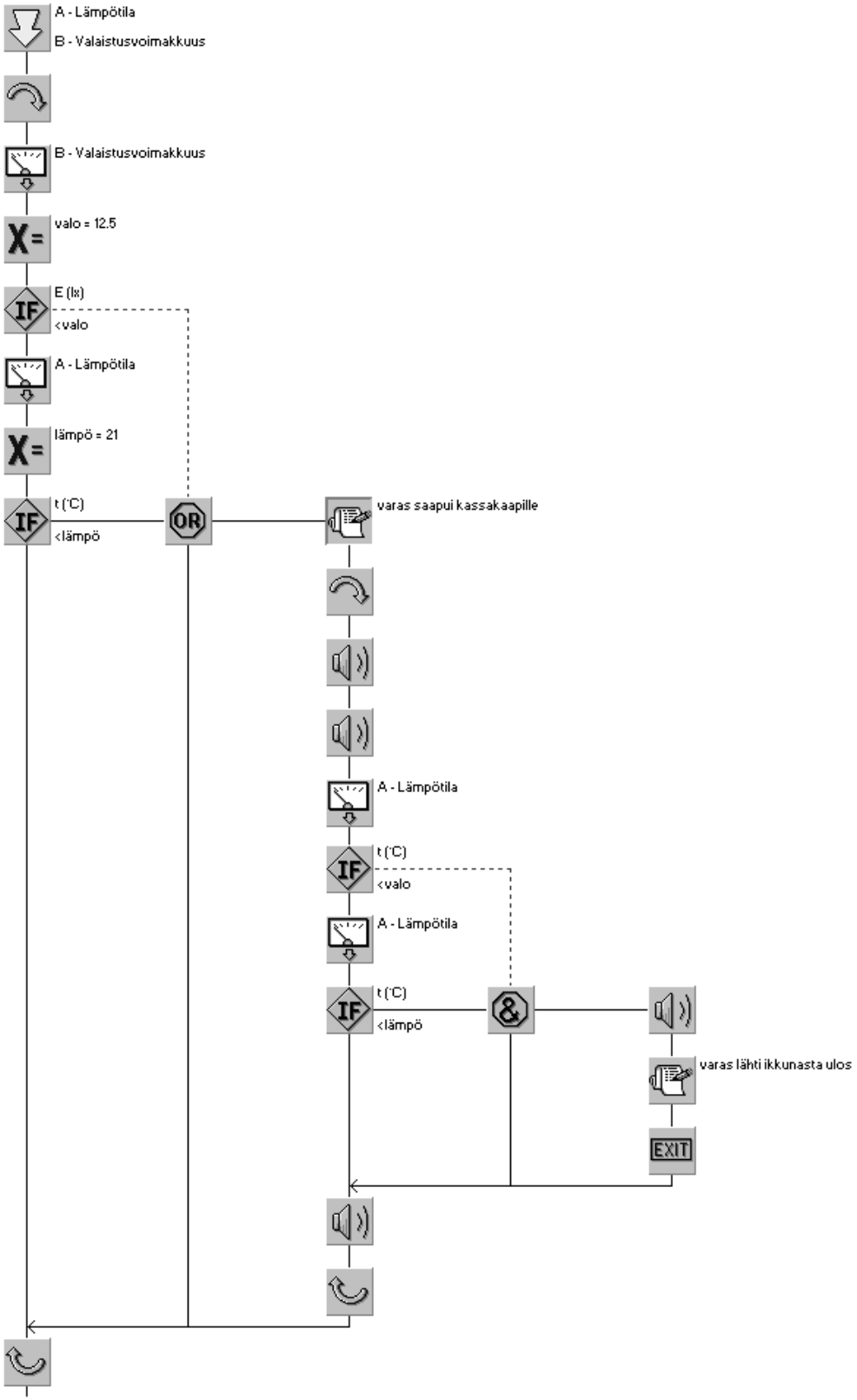
Liite 6: Lukio, Aloittelevien ryhmän videoanalyysi

Lukio - Aloittelevat	Ongelm. määr.					Mallintaminen alkaa:					2. pvä					
min	0-10	10-20	20-30	30-40	40-50	0-10	10-20	20-30	30-40	40-50	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80
kysymys opettajalle	2	3	2	0	3	14	2	2	0	1	2	2	1	0	3	1	1	0	7
kysym., idean johdattelua ryhmässä	0	0	3	1	2	22	7	4	1	0	9	2	1	0	2	1	3	3	33
keskustelu open kanssa	1	2	0	0	0	5	0	0	0	0	0	0	0	0	0	0	1	2	6
ideointi keskustelemalla	6	3	2	7	1	29	4	3	6	6	3	1	0	2	1	0	4	3	30
ideointi paperille	5	3	1	6	2	25	5	0	0	0	3	2	1	0	0	0	1	0	3
ideointi eleillä	3	1	5	4	1	16	2	0	0	0	0	1	0	0	0	0	0	0	0
opettaja äänessä (ohjeet, yleiset)	4	5	4	2	3	20	0	0	0	0	2	3	0	0	1	0	0	1	4
open kys. ryhmälle (johd., apu)	1	4	2	3	2	19	1	1	0	0	3	5	2	0	4	6	2	1	19
väittely	0	0	0	1	0	2	0	1	0	0	1	0	0	0	1	4	3	2	15
hiljaisuus	2	3	3	3	1	17	1	1	0	0	4	1	3	1	1	2	0	1	7
naurua	5	5	3	3	1	20	1	1	0	0	2	0	3	1	1	1	5	1	12
joku poistuu ryhmästä	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
välineiden tutkiminen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
ongelmia ryhmässä	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
ÄÄNESSÄ:																			
Poika 1	5	4	6	8	5	42	7	7	6	5	7	8	6	5	4	5	6	4	40
Tyttö 1	4	5	8	8	3	45	9	8	5	6	5	5	4	5	6	7	10	8	50
Tyttö 2	3	1	4	4	0	24	8	4	6	3	5	5	6	6	5	7	7	3	41
Tyttö 3	7	5	6	7	6	48	8	9	6	4	5	5	7	3	4	2	2	3	31
NEUVOO MUITA:																			
Poika 1	0	0	0	0	0	4	4	0	0	0	0	0	0	0	0	0	0	2	14
Tyttö 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	1	13
Tyttö 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	6
Tyttö 3	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	1	0	4
AKTIVOI MUITA:																			
Poika 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tyttö 1	0	0	1	0	0	3	0	0	0	0	1	0	0	0	0	0	0	0	0
Tyttö 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tyttö 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

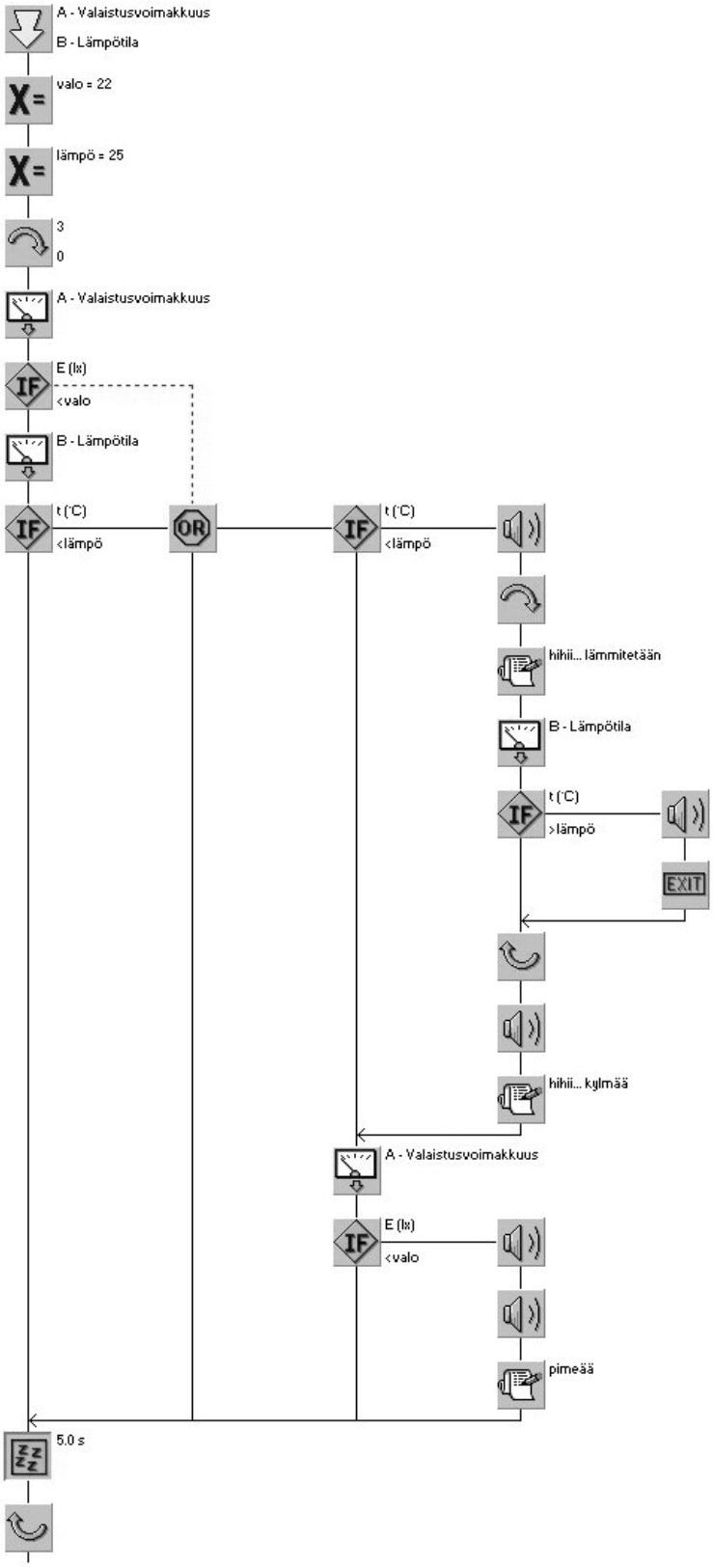
Liite 7: Lukio, Kokeneempien ryhmän videoanalyysi

Lukio - Kokeneemmat	Ongelm. määr.									Mallintaminen alkaa:									
min	0-10	10-20	20-30	30-40	40-50	0-10	10-20	0-10	2. pvä	10-20	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	
kysymys opettajalle	1	0	1	1	3	0	0	1	0	0	1	0	1	2	0	1	0	0	6
kysym., idean johdattelua ryhmässä	1	5	5	8	6	5	7	5	5	7	0	3	5	6	2	1	4	2	30
keskustelu open kanssa	0	1	1	1	2	1	0	0	1	0	0	0	0	1	0	1	3	1	7
ideointi keskustelemalla	5	5	4	5	5	4	5	3	4	5	0	0	2	2	1	2	1	1	11
ideointi paperille	3	6	2	7	0	2	2	0	2	2	0	0	0	1	0	0	0	0	1
ideointi eleillä	4	8	5	5	3	0	3	0	2	3	0	0	0	1	0	1	3	0	4
opettaja äänessä (ohjeet, yleiset)	3	0	3	2	0	1	1	0	1	0	1	0	0	1	0	0	1	0	9
open kys. ryhmälle (johd., apu)	0	2	3	1	0	0	2	0	0	1	0	0	0	1	0	0	0	0	7
väittely	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2
hiljaisuus	2	3	4	4	3	2	2	1	2	6	30	2	0	7	3	5	5	4	32
naurua	4	3	2	1	4	2	2	0	1	1	16	2	0	1	4	2	4	2	14
joku poistuu ryhmästä	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	3
välineiden tutkiminen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	19
ongelmia ryhmässä	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ÄÄNESSÄ:																			
Poika 1	10	12	10	11	7	8	11	7	8	11	69	7	6	8	7	7	6	5	52
Poika 2	2	6	5	2	5	10	7	5	0	3	23	1	1	0	0	1	0	0	2
Tyttö 1	7	8	11	11	10	7	7	8	8	7	62	8	7	6	9	5	6	6	57
Tyttö 2	7	6	7	10	7	7	7	7	7	7	51	7	4	4	7	3	6	2	35
NEUVOO MUITA:																			
Poika 1	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	1	0	8
Poika 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tyttö 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	2
Tyttö 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	1	0	5
AKTIVOI MUITA:																			
Poika 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Poika 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tyttö 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tyttö 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

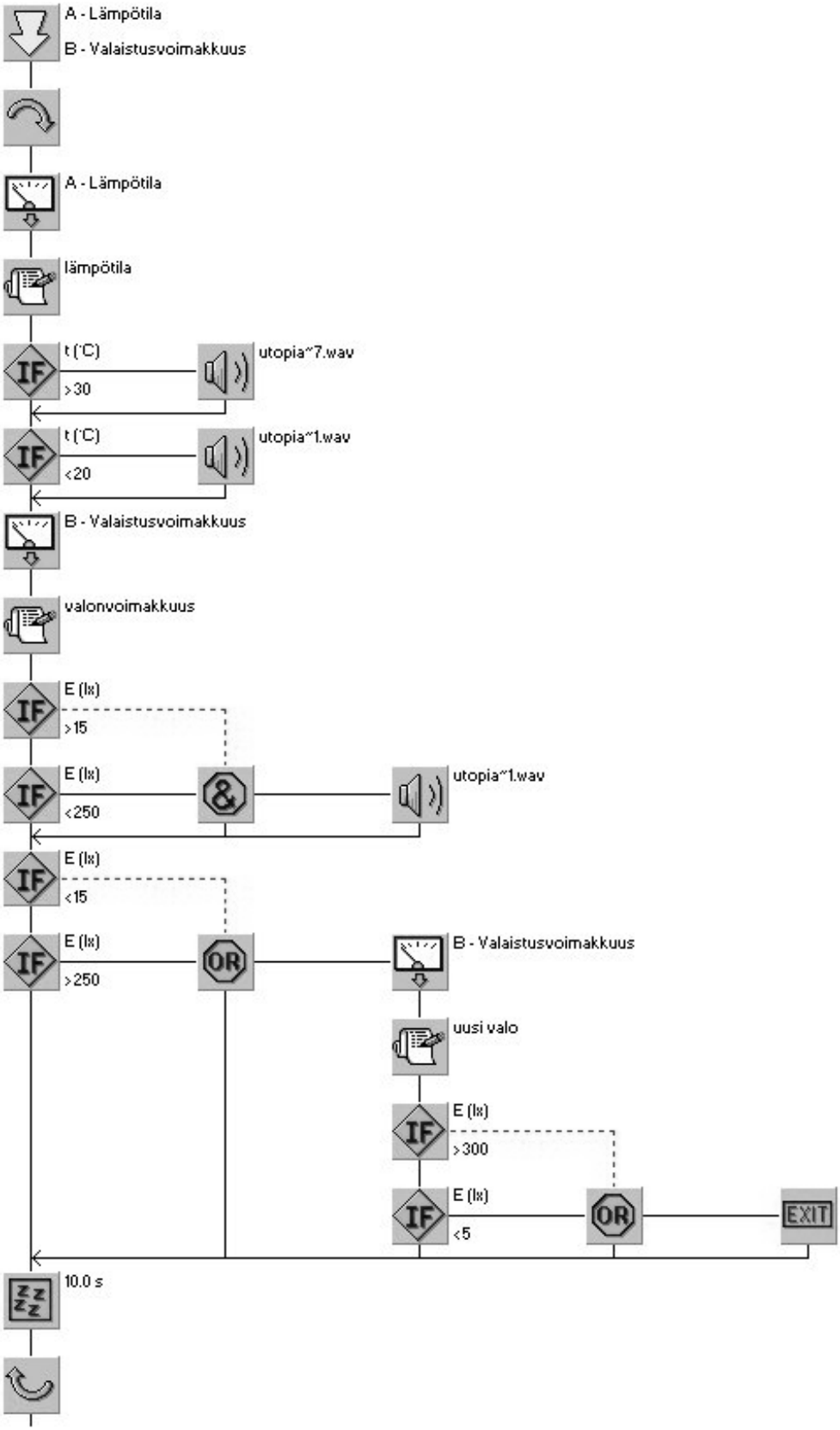
Liite 8: Aprikoosi-ryhmän valmis ohjelma



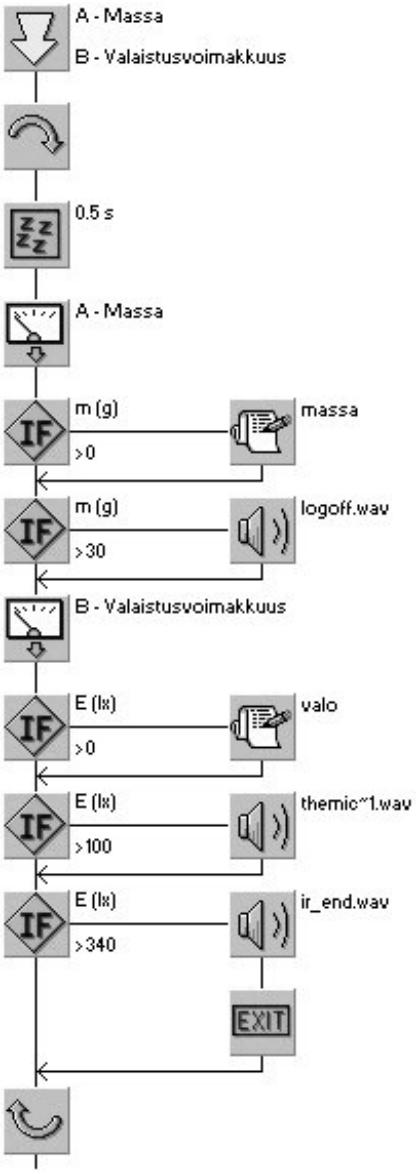
Liite 9: Fruitti-ryhmän valmis ohjelma



Liite 10: Aloittelevien ryhmän valmis ohjelma



Liite 11: Kokeneempien ryhmän valmis ohjelma



Liite 12: Ennakkokysely

Hei,
Seuraavassa kyselyssä tiedustelemme teidän kokemuksianne tietotekniikasta ja tietokoneiden käytöstänne. Kysymykset voivat tuntua hankalilta, mutta siitä ei kannata huolestua. Tämän tutkimuksen tarkoituksena on selvittää teidän tämänhetkiset kokemuksenne tietotekniikasta ja sen käytöstä koulussa.
Lisätietoja:

Harri Kähkönen
p. 050 345 9077
hkahkone@cs.joensuu.fi

Kai Piiroinen
p. 050 562 1877
kpiiroi@cs.joensuu.fi

Nimi: _____
Ikä: ___ Luokka: _____

1. Minulla / kotonani on tietokone.
□ Kyllä □ Ei
Jos ei, miksi?_____

2. Aion hankkia / kotiini hankitaan tietokone tulevaisuudessa.
□ Kyllä □ Ei
Jos ei, miksi?_____

3. Missä käytät tietokonetta?
□ Kotona □ Koulussa
□ Kaverini luona □ Kirjastossa
□ Muualla, missä: _____

4. Kuinka usein käytät tietokonetta?
□ Päivittäin □ Kerran viikossa
□ Kerran kuukaudessa □ Harvemmin
□ En ollenkaan, miksi?

_____
_____

5. Mihin käytät tietokonetta:
□ Pelaamiseen □ Kirjoittamiseen
□ Piirtämiseen □ Ohjelmoimiseen
□ Muuhun, mihin?

_____
_____

6. Jos olet ohjelmoinut tietokoneella, kerro hieman mitä ja miksi olet ohjelmoinut? Liittyikö se koulutyöhösi vai harrastuksiisi?

_____
_____

7. Kuinka paljon käytät tietokonetta koulussa tai koulutehtäviisi liittyen?

□ Päivittäin □ Pari kertaa viikossa
□ Kerran viikossa □ Kerran kuukaudessa
□ Harvemmin □ En ollenkaan

a) *Millaisia koulutehtäviä olet tehnyt tietokoneella?*
Aine: _____
Tehtävät:_____
_____
Aine: _____
Tehtävät:_____
_____
Aine: _____
Tehtävät:_____
_____
(jos tila ei riitä, pyydä lisäpaperia opettajalta)

b) *Millaiseksi koet tietokoneen käyttämisen koulutyössä?*
□ Pidän tietokoneen käyttämisestä □ En pidä tietokoneen käytöstä koulussa
□ Tietokoneesta on apua koulutyössä □ Tietokoneesta ei ole ollut minulle hyötyä

8. Käytetäänkö koulussa tarpeeksi tietokonetta?

□ Kyllä □ Ei

Jos ei, niin kuinka mielestäsi tietokoneen käyttöä tulisi lisätä?

_____
_____
_____

9. Millaista opetusta haluaisit lisää tietokoneiden osalta?

□ Ohjelmointi □ Kirjoittaminen / tekstinkäsittely
□ Piirtäminen / kuvankäsittely □ Lisälaitteiden asentaminen
□ Taulukkolaskenta □ Ohjelmien asentaminen
□ Muuta, mitä? _____
_____

10. Mihin käytät Internetiä ja tietoverkkoja yleensä?

□ Sähköposti □ Tiedonhaku
□ Viihde (pelit, chat yms.) □ Uutiset
□ Tiedostojen haku □ Lehtien lukeminen
□ Muuta, mitä?
_____
_____

11. Käytän internetiä ja tietoverkkoja:

□ Kotona □ Koulussa

□ Kirjastossa □ Kaverin luona
□ Muualla, missä?

_____
_____

12. Kuinka usein käytät internetiä ja tietoverkkoja?
□ Useita kertoja päivässä □ Päivittäin
□ Muutaman kerran viikossa □ Viikoittain
□ Pari kertaa kuussa □ Kerran kuussa
□ Harvemmin
□ En käytä, miksi?

_____
_____

13. Tietoverkkoja käytetään opetuksessa mielestäni
□ Liikaa □ Tarpeeksi
□ Liian vähän □ Ei ollenkaan

14. Tietoverkkojen käyttöä opetuksessa tulisi mielestäni
□ Lisätä □ Säilyttää ennallaan
□ Vähentää □ Poistaa kokonaan

a) Miksi olet tätä mieltä?

_____
_____

b) Missä tilanteissa olet tarvinnut tietoverkkoja koulussa?

_____
_____

15. Seuraavassa kysytään ohjelmointiin liittyviä asioita. Vastaa niin moneen kuin kykenet. Kysymykset voivat tuntua vaikeilta, mutta yhtä oikeaa vastausta ei ole.

a) Mitä ohjelmointi mielestäsi on?

_____
_____

b) Mikä on ohjelma?

_____
_____

c) Mitä tarkoittaa:
”silmukka”:

_____
_____

”ehtolause”:

_____

_____

”muuttuja”:

_____

_____

d) Mikä on ohjelmointikieli?

_____

_____

Liite 13: Loppukysely

Hei,
Kokeilu on nyt saatu päätökseen. Kiitokset siitä Sinulle! Pyytäisimme vielä, että täyttäisit tämän kyselylomakkeen huolellisesti ja rehellisesti. Mitään tietoja ei tulla luovuttamaan nimelläsi varustettuna eteenpäin. Eli kaikki vastaukset ovat luottamuksellisia ja ne tullaan käsitelemään sen mukaisesti.
Jos sinulle tulee vielä jotakin mieleen, niin kirjoita paperin kääntöpuolelle huomiosi.

Kiittäen,

Harri Kähkönen
p. 050 345 9077
hkahkone@cs.joensuu.fi

Kai Piiroinen
p. 050 562 1877
kpiiroi@cs.joensuu.fi

Nimi: _____
Ikä: ___ Luokka: _____

1. Aion hankkia / kotiini hankitaan tietokone tulevaisuudessa.
□ Kyllä □ Ei
Jos ei, miksi?_____

2. Pidin tästä kurssista?
□ Kyllä □ En

Perustele vastauksesi
_____
_____

3. Kurssilla käsitellyt asiat olivat mielestäni mielenkiintoisia
□ Kyllä □ Ei

Perustele vastauksesi
_____
_____

4. Uskon että osallistumisestani Empirica Control -kokeiluun oli minulle hyötyä
□ Ei □ Ehkä vähän
□ Kyllä

Perustele vastauksesi
_____
_____

5. Aion ottaa jatkossa ohjelmointiin liittyviä kursseja
□ En □ Kyllä

Vaikuttiko kokeilu päätökseesi, ja miten (tai miksi ei)?

_____

_____

6. Pitäisikö tällaisia kokeiluja mielestäsi lisätä kouluissa?

□ Ei □ Ehkä vähän

□ Paljon

Perustele vastauksesi

_____

_____

7. Miten projekti muutti suhtautumistasi ohjelmointia kohtaan?

_____

_____

_____

8. Oliko Empirica Control mielestäsi (perustele vastauksesi)

Selkeä käyttää

_____

_____

Mielenkiintoinen

_____

_____

Hyödyllinen ohjelmoinnin oppimisen kannalta

_____

_____

d) Muita huomioita?

_____

_____

9. Millaiseksi koit ryhmäsi työskentelyn? (perustele vastauksesi)

Osallistuivatko kaikki tasapuolisesti työskentelyyn?

_____

_____

Keskusteltiinko esiintyvistä ongelmista yhdessä?

_____

_____

Kuinka ongelmat ratkaistiin (kompromisseja tms.)?

_____

_____

Auttoiko ongelman tarkka mallintaminen ratkaisun tekemisessä?

Olisiko ryhmän toimintaa voinut mielestäsi tehostaa ja miten?

Parhaat ja huonoimmat muistosi ryhmän työskentelystä:

10. Seuraavassa kysytään kertauksen vuoksi ohjelmointiin liittyviä asioita. Muista, että yhtä oikeaa vastausta ei ole.

a) Mitä ohjelmointi mielestäsi on?

b) Mikä on ohjelma?

c) Mitä tarkoittaa:
”silmukka”:

”ehtolause”:

”muuttuja”:

d) Auttoiko Empirica Control mielestäsi hahmottamaan ohjelmointia?

e) Mitä hyviä ja huonoja puolia Empirica Controlissa oli?

Hyvät:

Huonot:

Liite 14: Empirica Control 1.1a -opetusmateriaali

Empirica Control 1.1a -oppimateriaalissa olevat kuvat ja videoleikkeet ovat tekijöiden valmistamia. Oppimateriaali toimii web-selaimissa jotka tukevat kehyksiä. Materiaalin sisältämät videoleikkeet ovat avi-tiedostoja joita esimerkiksi Windows Media Player-ohjelmisto toistaa.

Oppimateriaalin voi käynnistää avaamalla *index.htm*-tiedoston selaimeen. Empirica-mittausjärjestelmän toimintakuntoon saattamiseksi löytyy tietoja tiedostosta *getstart.htm*.

Harri Kähkönen
Kai Piiroinen