Empiirinen koe PlanAnin kuvallisista metaforista

Tuija Stützle

22.6.2004

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

Tiivistelmä

Muuttujan rooli kuvaa sitä miten muuttuja saa peräkkäiset arvonsa ja mikä sen suhde on muihin muuttujiin. Noviisitason proseduraalisen ohjelmoinnin rooleja on löydetty yhteensä kymmenen: kiintoarvo, askeltaja, tuoreimman säilyttäjä, sopivimman säilyttäjä, kokooja, muuntaja, seuraaja, yksisuuntainen lippu, järjestelijä ja tilapäissäilö. PlanAni on ohjelma-animaattori, jolla animoidaan pieniä ja yksinkertaisia ohjelmia muuttujien rooliteoriaan perustuen. Jokaista muuttujaa havainnollistetaan sen roolia esittävällä kuvalla ja muuttujan saamia arvoja sekä muuttujan vertailua havainnollistetaan rooliin kuuluvilla animaatioilla. Metaforatutkimuksen mukaan oikein valittu metafora helpottaa asian ymmärtämistä ja stimuloi ajatusprosesseja. Tässä tutkimuksessa on kartoitettu miten PlanAnissa rooleista käytetyt kuvalliset metaforat välittävät roolien ominaisuuksia ja näin edistävät oppimista. Tutkimuksen kohteena oli viisi roolia: kiintoarvo, askeltaja, seuraaja, kokooja ja tilapäissäilö. Kokeen suoritti kaksi koeryhmää, joista rooliryhmä sai arvioitavaksi PlanAnissa käytetyt alkuperäiset roolikuvat ja kontrolliryhmä koetta varten keksityt lumekuvat. Kokeella kerättiin tietoa metaforien ominaisuuksista ennen kuin kuvat esiteltiin PlanAnilla, mitattiin kuinka hyvin koehenkilöt tunnistavat roolit kuvista PlanAnilla animoidun ohjelman suorituksen jälkeen, pyydettin koehenkilöiden arvioita eri kuville sekä vaihtoehtoja rooleja paremmin esittäviksi kuviksi, mitattiin rooliteorian omaksumista ennen ja jälkeen PlanAnilla suoritettua animointia sekä kyseltiin koehenkilöiden mielipiteitä roolikuvien tarpeellisuudesta. Tulosten mukaan kiintoarvoa, askeltajaa ja kokoojaa esittävät alkuperäiset roolikuvat välittävät roolien ominaisuuksia oppilaille lumekuvia paremmin. Seuraajan ja tilapäissäilön alkuperäiset roolikuvat eivät tulosten perusteella ole lumekuvia parempia metaforia rooleilleen.

ACM-luokat (ACM Computing Classification System, 1998 version): H.1.2, D.2.2, K.3.2

Avainsanat: muuttujien roolit, animointi, visualisointi, ohjelmointi, metafora, empiirinen tutkimus, ohjelmoinnin psykologia, ohjelmoinnin opettaminen

Esipuhe

Lähdin neljänkympin kriisin tuloksena opiskelemaan uudelleen tietojenkäsittelytiedettä. Takana oli atk-instituutin datanomitutkinnon lisäksi parikymmentä vuotta alan töitä erilaisissa yrityksissä ja myös itsenäisenä yrittäjänä. Päälimmäisenä tunteena oli: tätäkö tämä nyt sitten on seuraavat 20 vuotta? No ainakin parin vuoden opiskelu toisi hieman muutosta puuduttavaan arkeen.

Opiskelurutiinin löytymisen jälkeen kurssi seurasi toista ja asiat tuntuivat tutuilta ja melko helpoilta omaksua. Sitten tuli ohjelmoinnin empiirisen tutkimuksen kurssi, joka teki minuun suuren vaikutuksen ja sai kiinnostumaan tutkimuksesta. Halusin kokeilla miltä oikean tutkimuksen teko tuntuu. Gradu oli pyörinyt mielessä lähinnä isona ja pelottavana peikkona ja halusin varmistaa, että saan mielenkiintoisen aiheen jonka avulla jaksan taistella peikkoa vastaan. Siispä keräsin rohkeuteni ja marssin professori Jorma Sajaniemen juttusille sopivan tutkimuskohteen löytämiseksi.

Kiinnostuin välittömästi Sajaniemen kehittämästä muuttujien rooliteoriasta ja halusin olla mukana sen tutkimuksessa. Alussa oli vaikeuksia löytää sopivaa metaforakirjallisuutta, koska sitä on kirjastot pullollaan aiheina kielitiede, runous, musiikki jne. Sajaniemi opasti minut oikeille lähteille, joita ei olisi kirjastosta löytynytkään. Puoli vuotta kestäneen sulattelun jälkeen aloin kirjoittaa gradua ja samalla suunnitella koetta. Työ oli mielenkiintoista ja erittäin haastavaa ja siinä sivussa sain gradun valmiiksi. Suurin ongelma oli koehenkilöiden suostuttelu, mutta lopulta sain heitäkin tarpeeksi tilastokelpoisen aineiston saamiseksi.

Haluaisin kiittää professori Sajaniemeä kärsivällisestä ja rakentavasta ohjauksesta, Pirkko Pölöstä jatkuvasta kannustuksesta sekä avusta kokeen järjestämisessä, Ville Rapaa PlanAnin muutoksista sekä teknisestä tuesta, Eija Mikkosta sekä Marja Valkosta kokeen esitestauksesta ja varsinaisia koehenkilöitä kokeeseen osallistumisesta. Peikko on voitettu.

Sisältö

1 Johdanto

Ohjelmoimaan oppiminen on monelle vaikeaa ja tätä helpottamaan yritetään löytää erilaisia apuvälineitä ja keinoja. Sajaniemi (2002a) on kehittänyt teorian muuttujien rooleista, joiden on todettu helpottavan ohjelmoinnin alkeiden oppimista ja parantavan oppimistulosta (Sajaniemi & Kuittinen, in press). Muuttujan rooli kuvaa sitä, miten muuttuja saa peräkkäiset arvonsa ja mikä sen suhde on muihin muuttujiin. Erilaiset muuttujien roolit löytyivät analysoimalla kolmen aloittelijoille tarkoitetun Pascal-oppaan esimerkkiohjelmia: kiintoarvo, askeltaja, tuoreimman säilyttäjä, sopivimman säilyttäjä, kokooja, muuntaja, seuraaja, yksisuuntainen lippu, järjestelijä ja tilapäissäilö. Kiintoarvo on muuttuja, jonka arvoa ei muuteta alustuksen jälkeen. Tyypillisesti kiintoarvo on syötteenä annettu tieto, joka talletetaan muuttujaan ohjelman suorituksen ajaksi. Askeltaja saa arvonsa jollakin ennustettavalla tavalla, esimerkiksi taulukon läpikäynnissä toimiva indeksi on askeltaja. Tuoreimman säilyttäjä säilyttää viimeisimmän käsiteltävänä olevasta sarjasta arvoja, esimerkiksi viimeksi luettu syöte voidaan tallettaa tuoreimman säilyttäjään. Jokaisella roolilla on oma tyypillinen käyttäytymisensä, joka opetetaan aloittelevalle ohjelmoijalle sitä mukaa kuin ne esiintyvät esimerkkiohjelmissa. Muuttujien roolien opettamisen yhteydessä oppilaille annetaan myös strategista tietoa roolien käytöstä ohjelmissa. Roolit käsitteinä helpottavat ohjelmoinnin periaatteiden selittämistä oppilaille.

Ohjelmakoodia havainnollistamaan on kehitetty myös ohjelma-animaattori, PlanAni (Sajaniemi & Kuittinen, 2003), jolla rooliteoriaan perustuen animoidaan yksikertaisten ohjelmien suoritusta. Jokaista muuttujan roolia esittää kuva, jolla pyritään välittämään käyttäjälle muuttujan roolin tärkeimpiä ominaisuuksia. Esittelen rooliteorian ja siihen perustuvan PlanAni-ohjelma-animaattorin luvussa 2.

Tässä pro-gradu-tutkielmassani olen empiirisellä kokeella tutkinut PlanAnissa käytettyjä roolikuvia ja niiden sopivuutta esittämään kyseistä roolia. Olen perustanut tutkimukseni metaforateoriaan ja käyttänyt tutkimusmenetelmänä Alty *& al:*in (2000) metaforan valintaan, toteutukseen ja arviointiin kehittämää ohjeistoa, joka on kehitetty käytännön kokemuksiin nojaten ja viimeistelty useiden eri maissa toimivien ohjelmistotuotannon ammattilaisten ryhmätyönä. Luvussa 3 kerron metaforateoriasta ja sen soveltamisesta tietojärjestelmissä sekä esittelen käyttämäni tutkimusmenetelmän.

Empiirinen koe kartoitti viiden muuttujan roolin ominaisuuksia: askeltaja, kiintoarvo, seuraaja, kokooja ja tilapäissäilö. Kaikkia rooleja ei otettu mukaan tutkimukseen, koska koetilaisuus olisi silloin venynyt kohtuuttoman pitkäksi ja koehenkilöiden saatavuus entisestäänkin vaikeutunut. Lisäksi etukäteen ei voitu tietää kuinka hyvin koejärjestely toimii ja saadaanko sillä esille eroja kuvien välillä. Koska koejärjestely osoittautui toimivaksi näillä viidellä ensimmäisellä roolilla, voidaan tutkimusta jatkaa tekemällä sen pohjalta koe viimeisten roolikuvien testaamiseksi.

Koetta varten kehitin rooleille myös ns. lumekuvat, jotka annettiin kontrolliryhmän tutkittaviksi; rooliryhmä arvioi alkuperäisiä roolikuvia. Kokeella halusin selvittää heijastavatko alkuperäiset roolikuvat roolien ominaisuuksia paremmin kuin lumekuvat, tunnistetaanko alkuperäiset roolikuvat PlanAnin käytön jälkeen paremmin kuin lumekuvat sekä mitä mieltä koehenkilöt ovat kuvista. Lisäksi halusin saada koehenkilöiden omia ehdotuksia rooleja paremmin esittäviksi kuviksi sekä palautetta siitä onko roolien esittäminen kuvilla ylipäätään tarpeellista. Esittelen luvussa 4 koehenkilöt ja tehtävät sekä kerron kokeen käytännönjärjestelyistä, tuloksista ja niiden arvioinnista. Lopuksi on yhteenveto luvussa 5.

2 Muuttujien roolit

Sajaniemi ja Kuittinen (in press) ovat kehittäneet uuden tavan opettaa ohjelmoinnin alkeita aloittelijoille. Teorian mukaan ohjelmoinnin alkeiden oppiminen helpottuu ja tulos paranee, kun oppilaille esitetään esimerkkiohjelmia selittäen niitä muuttujan roolikäsitteen avulla. Oppilaille saadaan roolipohjaisen opetuksen avulla välitettyä niin sanottua hiljaista tietoa, jota yleensä syntyy vasta pidemmän ohjelmointikokemuksen tuloksena. Lisäksi ohjelman visualisoinnin PlanAni-ohjelma-animaattorin avulla on todistettu antavan oppilaille syvempää ymmärrystä ohjelman sisällöstä kuin pelkkä rooleihin perustuva opetus ilman visualisointia.

Tässä luvussa esittelen ensin erilaiset muuttujien roolit ja niiden ominaisuudet. Seuraavaksi kerron muuttujien rooliteoriaan perustuvasta ohjelmakoodin visualisoinnista PlanAnilla ja lopuksi esittelen tutkimustietoa PlanAnin käytöstä ohjelmoinnin alkeiden opetuksessa.

2.1 Muuttujan rooli käsitteenä

Muuttujan rooli (the role of variable) kuvaa sitä, miten muuttuja saa peräkkäiset arvonsa ja mikä sen suhde on muihin muuttujiin (Sajaniemi, 2002a). Rooli ei siis kuvaa sitä, mihin tarkoitukseen muuttujaa käytetään. Kuvassa 1 olevassa ohjelmassa on käytetty useita rooleiltaan erilaisia muuttujia.

Muuttuja fib saa alkuarvokseen ensimmäisen Fibonacci-luvun eli ykkösen. Sen jälkeen fib-muuttujaan lasketaan aina seuraava Fibonacci-luku summaamalla kaksi aiempaa Fibonacci-lukua yhteen. Tämänkaltaiselle muuttujalle on Sajaniemen teoriassa (2002a) annettu rooli *kokooja* (gatherer). Lastfib-muuttuja saa aina arvokseen fib-muuttujan vanhan arvon eli se ikäänkuin seuraa fib-muuttujaa ja on siis roolinimeltään *seuraaja* (follower). Jotta saataisiin seuraava Fibonacci-luku laskettua, täytyy viimeistä edellinen Fibonacci-luku tallettaa tilapäisluonteiseen muuttujaan temp, jonka roolina on *tilapäissäilö* (temporary). Number-muuttuja saa vain yhden arvon ja säilyttää sen kiinteästi koko ohjelman suorituksen ajan, joten se on rooliltaan *kiintoarvo* (fixed value). Muuttuja i saa arvoja kolmosesta lähtien number-muuttujan sisältämään arvoon saakka kasvaen aina yhdellä suuremmaksi. Tällaista ennalta tiedetyllä numerosarjalla arvotettua muuttujaa kutsutaan roolinimellä *askeltaja* (stepper).

2.2 Muuttujien roolit ja niiden ominaisuudet

Sajaniemen (2002a) mukaan erilaiset muuttujien roolit löytyivät analysoimalla kolmen aloittelijoille tarkoitetun Pascal-oppaan esimerkkiohjelmia. Näin löytyneet yhdeksän roolia kattoivat 99 prosenttia muuttujista. Myöhemmin (Ben-Ari & Sajaniemi, 2004) löytyi vielä yksi rooli lisää, muuntaja. Näin on syntynyt seuraava roolien joukko.

Kiintoarvo: Kiintoarvo on muuttuja, jonka arvoa ei muuteta alustuksen jälkeen. Ohjelmassa voi olla useampia erilaisia alustuksia muuttujalle, mutta alustuksen jälkeen sen arvo ei muutu. Esimerkkinä kiintoarvosta voisi olla syötteenä annettu tieto, joka talletetaan muuttujaan ohjelman suorituksen ajaksi.

```
program fibonacci;
var lastfib,fib,temp,number,i:integer;
begin
   lastfib := 1;fib := 1;
   if number <= 2 then
       writeln('Kaksi ensimmäistä ovat kumpikin 1.')
   else begin
       writeln('1.luku on 1');
       writeln('2.luku on 1');
       for i := 3 to number do begin
           temp := lastfib;
           lastfib := fib;
           fib := fib + temp;
           writeln(i:2,'.luku on ',fib)
       end
   end
end.
```

Kuva 1: Fibonacci-ohjelma Pascalilla.

Askeltaja: Askeltaja on muuttuja, joka saa arvonsa jollain ennustettavalla tavalla. Esimerkkinä askeltajasta voisi olla muuttuja, joka toimii indeksinä taulukon läpikäynnissä tai muuttuja, jolla lasketaan syötteiden lukumäärä.

Tuoreimman säilyttäjä: Tuoreimman säilyttäjä on muuttuja, joka säilyttää viimeisimmän käsiteltävänä olevasta sarjasta arvoja. Esimerkkinä voisi olla viimeksi luettu taulukon alkio, kun ollaan käymässä taulukkoa läpi askeltajan avulla. Tuoreimman säilyttäjän arvo voidaan myös muuntaa toisenlaiseen talletusmuotoon heti arvon asetuksen jälkeen. Myös viimeksi luettu syöte voidaan tallettaa tuoreimman säilyttäjään.

Sopivimman säilyttäjä: Sopivimman säilyttäjään talletetaan jollakin tavalla mitattu paras arvo tietystä käsiteltävänä olevasta sarjasta. Muuttuja voi vaihtaa arvoaan useita kertoja sarjan läpikäynnin aikana, koska myöhemmin voi löytyä vielä parempi arvo. Esimerkiksi ohjelma voisi hakea syötesarjan pienintä arvoa, jolloin sen talletukseen käytetty muuttuja olisi rooliltaan sopivimman säilyttäjä.

Kokooja: Kokooja saa arvonsa summaamalla tai kumuloimalla muiden muuttujien arvoja. Esimerkkinä kokoojasta voisi olla syötteinä annettujen kokonaislukujen yhteissumman laskuri.

Muuntaja: Muuntaja saa aina arvonsa toisten muuttujien arvojen laskennan tuloksena. Esimerkkinä muuntajasta voisi olla muuttuja, johon lasketaan syötteinä annettujen kokonaislukujen keskiarvo aina uuden syötteen jälkeen. Syötteenä annettujen lukujen summa lasketaan kokoojana toimivaan summa-muuttujaan ja syötteiden lukumäärän laskee askeltajan roolissa toimiva lkm-muuttuja. Keskiarvo saa arvonsa laskutoimituksesta summa / lkm.

Seuraaja: Seuraajan roolissa oleva muuttuja saa uudeksi arvokseen aina jonkin toisen muuttujan vanhan arvon. Esimerkkinä on listan läpikäynti, jossa seuraajana toimivaan muuttujaan on talletettu käsiteltävää alkiota edeltävän alkion tiedot. Käsiteltävän alkion tiedot on talletettu muuttujaan, joka toimii tuoreimman säilyttäjänä.

Yksisuuntainen lippu: Yksisuuntainen lippu on Boolen muuttuja, joka voi saada vain 2 arvoa: true ja false. Yksisuuntainen lippu ei voi saada enää alkuperäistä arvoaan sen jälkeen kun se on muuttunut. Esimerkkinä yksisuuntaisen lipun käytöstä on esimerkiksi ohjelma, jossa sillä tarkkaillaan esiintyykö syötteiden joukossa negatiivisia

arvoja. Jos yksikin negatiivinen arvo löytyy, saa lippu arvon true eikä se enää koskaan voi saada takaisin alustusarvoaan false.

Järjestelijä: Järjestelijä on taulukko, jota käytetään alkioiden uudelleen järjestämiseen. Esimerkiksi järjestelijä taulu alustetaan joillakin syötetyillä merkeillä, jonka jälkeen merkkien järjestys vaihdetaan päinvastaiseksi.

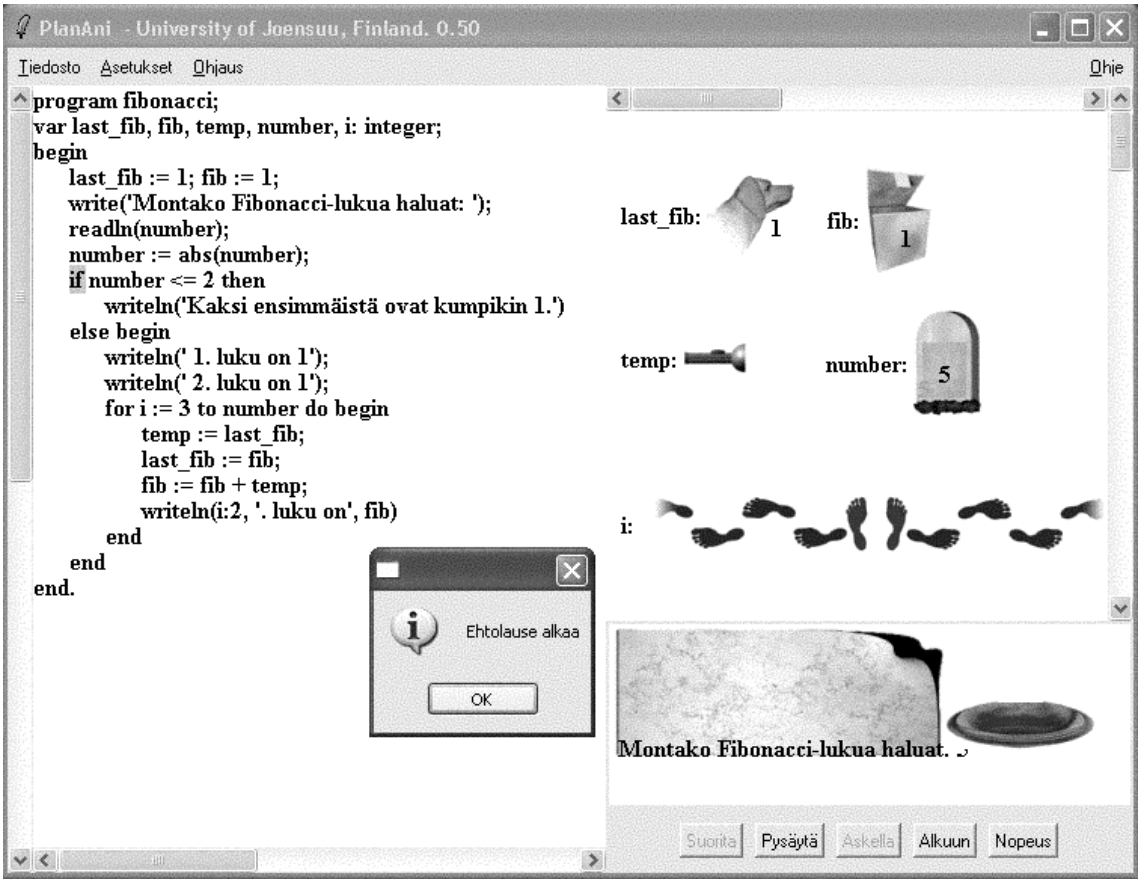
Tilapäissäilö: Tilapäissäilö tallettaa jonkin toisen muuttujan arvon vähäksi aikaa. Esimerkkinä voisi olla swap-operaatiossa käytetty muuttuja lajittelussa.

2.3 Ohjelman visualisointi PlanAnilla

PlanAni (Sajaniemi & Kuittinen, 2003) on opetuskäyttöön tarkoitettu ohjelma-animaattori, jolla voidaan visualisoida pieniä ohjelmia. PlanAnin animointi perustuu erilaisiin muuttujien rooleihin, joita esittämään on valittu kuvat. PlanAnia on käytetty ohjelmoinnin perusteiden opetuksessa aloittelijoille ja sen on todettu parantavan oppimistulosta (Sajaniemi & Kuittinen, in press). Kuvassa 2 nähdään PlanAni visualisoimassa yksinkertaista ohjelmaa.

Ohjelmakoodi on sijoitettu ikkunan vasempaan puoliskoon ja kulloinkin visualisoitavana oleva kohta korostetaan värillä. Ikkunan oikea yläpuoli on varattu muuttujien visualisointiin ja oikea alapuoli toimii syöte- ja tulostusalueena. Syötteen paikkaa kuvaa lautanen ja tulosteet tulevat paperilapulle. Kulloinkin aktiivisena oleva ohjelman kohta vasemmalla on yhdistetty nuolella vastaavaan muuttujaan oikealla.

PlanAni kertoo tekemisistään etukäteen ikkunan keskelle tulostuvaan ilmoitusikkunaan. Käyttäjän täytyy aina kuitata ilmoitus vastaanotetuksi ennen animoinnin jatkumista. Tämä helpottaa käyttäjää kohdistamaan huomionsa oikeisiin asioihin näytöllä. PlanAnin myöhemmissä versioissa ilmoitusikkuna on tarkoitus korvata puhesyntetisaattorilla. Käyttäjä voi säätää animoinnin nopeutta ja ikkunoissa käytettyä kirjasimen kokoa. Animoinnin voi koska tahansa aloittaa alusta, mutta ohjelmaa ei voi animoida takaisinpäin.



Kuva 2: Ohjelman visualisointia PlanAnilla.

2.4 Muuttujan roolien visualisointi

Roolin visuaalisen esitystavan piti Sajaniemen (2002b) mukaan olla sekä yleinen että erityinen. Yleinen siinä mielessä, että se kuvaisi kaikkia samassa roolissa olevia muuttujia ja erityinen tavalla, joka kuvaisi kyseisessä roolissa olevan muuttujan peräkkäisiä arvoja ja niiden suhdetta toisiinsa sekä muihin muuttujiin. Esimerkiksi kiintoarvoroolin visualisoinnin piti ilmaista, että muuttujan arvo ei ole muutettavissa.

Taulukossa 1 on listattu kaikkien roolien ominaisuudet, joita vastaavat kuvalliset esitykset Sajaniemi (2002b) pyrki löytämään.

Taulukko 1: Roolien visualisoitavat ominaisuudet (Sajaniemi, 2002b).

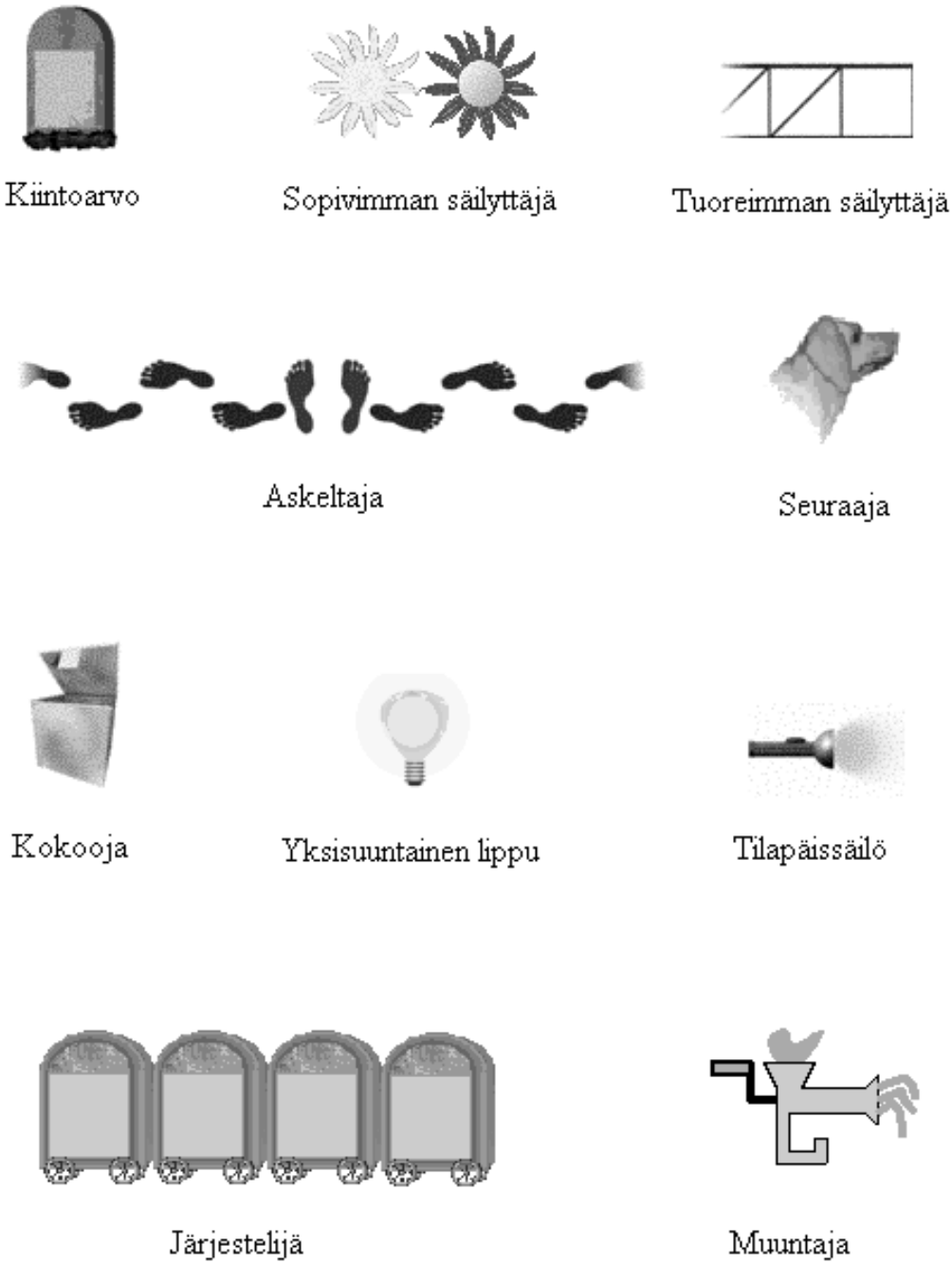
Rooli	*Ominaisuudet*
Kiintoarvo	Mahdotonta muuttaa.
Askeltaja	Tulevat arvot voidaan ennustaa, jos aikaisemmat arvot tiedetään. Yleensä arvot muuttuvat johonkin suuntaan: kasvavat tai pienenevät.
Tuoreimman säilyttäjä	Arvot saadaan sarjasta käsiteltäviä tietoja, mutta ne voivat olla mitä tahansa eli arvoilla ei ole kiinteää suhdetta keskenään.
Sopivimman säilyttäjä	Nykyinen arvo on parempi kuin mikään muu aikaisemmista arvoista.
Kokooja	Uusi arvo saadaan, kun yhdistetään vanha arvo uuden tiedon kanssa.
Muuntaja	Uusi arvo saadaan muokkaamalla muista muuttujista.
Seuraaja	Kiinteästi yhteydessä toiseen muuttujaan, yleensä sen edellinen arvo.
Yksisuuntainen lippu	Vain kaksi mahdollista arvoa, mahdotonta muuttaa takaisin alkuarvoonsa.
Järjestelijä	Yksittäisiä osia ei voi muuttaa, mutta niitä voi siirtää paikasta toiseen.
Tilapäissäilö	Hyvin lyhytaikainen.

Kuvassa 3 esitetään rooleja vastaavat visualisoinnit. Kiintoarvoa kuvaa hautakivi, johon hakattu tieto ei ole helposti muutettavissa (Sajaniemi & Kuittinen, 2003). Sopivimman säilyttäjää kuvataan erivärisillä kukkasilla, joista kirkasvärinen kuvaa muuttujan nykyistä eli sopivinta siihen mennessä löytynyttä arvoa. Harmaa kukkanen kuvaa muuttujan edellistä eli toiseksi sopivinta arvoa. Tuoreimman säilyttäjää esittävä kuva näyttää myös viimeistä edellisen arvonsa, mutta koska tässä tapauksessa kumpikaan arvoista ei ole toistaan parempi, ne esitetään samanlaisten, neutraalien laatikoiden sisällä. Muuttujan vanha arvo on vain yliviivattu, jotta nykyinen arvo erottuu.

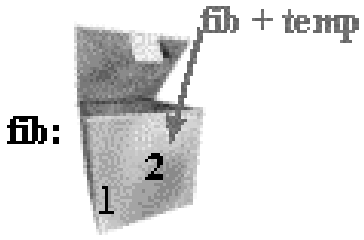
Askeltajaa kuvaavat jalanjäljet, jotka ovat seisahtuneet muuttujan nykyisen arvon kohdalle. Kuvassa näkyy myös askeltajan mahdolliset seuraavat ja edelliset arvot. Askelten kulkusuuntaa eli muuttujan saamia seuraavia arvoja kuvataan nuolella. Seuraajaa kuvaa koira, joka istutetaan sen muuttujan perään, jota se seuraa. Kokoojaa kuvaa laatikko, jossa näkyy sekä keskellä oleva nykyinen arvo että vasemmassa alalaidassa oleva edellinen arvo. Yksisuuntaista lippua kuvaa hehkulamppu silloin, kun muuttujan alkuarvo on true. Hehkulamppu särkyy lipun saatua false-arvon. Jos yksisuuntaisen lipun alkuarvo on false, sitä kuvataan kananmunalla. Kanapoika kuoriutuu, kun lippu saa true-arvon. Tilapäissäilöä kuvaa taskulamppu, joka on päällä juuri niin kauan kuin sitä tarvitaan. Kun tilapäissäilön arvo tulee tarpeettomaksi, taskulamppu sammuu ja arvo häviää.
Taulukoiden elementtejä kuvataan elementtiä kuvaavalla roolilla. Jos esimerkiksi taulukko sisältää kokoojia, se kuvataan sarjana laatikoita. Järjestelijä-rooli kuvataan siten kuin taulukon elementit olisivat kiintoarvoja, joiden alle on helpompaa liikuttelua varten laitettu pyörät.

2.5 Roolipohjainen animointi PlanAnissa

Muuttujien rooleja esittävien kuvien lisäksi myös muuttujille tehtävien operaatioiden animointi on roolipohjaista (Sajaniemi & Kuittinen, 2003). Muuttujan alustaminen, vertailu ja uuden arvon asettaminen voi olla erilaista eri rooleilla, koska siinäkin on pyritty ilmentämään kyseisen roolin erityisiä ominaisuuksia.



Kuva 3: Muuttujien rooleja esittävät kuvat.



Kuva 4: Uusi arvo kokoojan roolissa olevalle muuttujalle.

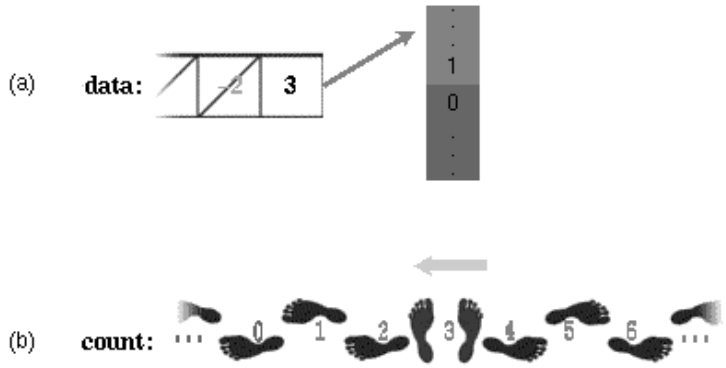
Esimerkiksi kuvassa 4 kokoojan roolissa oleva muuttuja saa uuden arvon lauseella "fib := fib+temp" siten, että sen vanha arvo siirtyy kuvan vasempaan alalaitaan ja muuttuu harmaaksi. Sitten "fib+temp" -lause ilmestyy kuvan oikealle yläpuolelle osoittaen nuolella kuvan keskelle, johon ilmestyy lauseen mukainen uusi arvo. Lopuksi laskentalause ja nuoli häviävät. Animaatio ilmentää kokoojan roolissa olevan muuttujan luonnetta näyttämällä miten uusi arvo saadaan lisäämälle vanhaan arvoon jotain uutta.

Kuva 5: Uusi arvo askeltajan roolissa olevalle muuttujalle.

Askeltaja puolestaan saa uuden arvonsa ihan eri tavalla, kuten kuvassa 5 näkyy. Koska jo alustusvaiheen jälkeen askeltajan seuraavat ja edelliset arvot ovat kuvassa nähtävillä ja nuoli on osoittamassa arvojen kulkusuuntaa, siirtyvät kaikki askeltajan nykyiset ja entiset arvot yhden position vasemmalle muuttujan kasvaessa tai yhden position oikealle muuttujan pienentyessä siten että uusi arvo jää seisahtaneiden jalkojen kohdalle.

Kuvassa 6 on esitetty kahden erilaisessa roolissa olevan muuttujan vertailuanimaatio "muuttuja > 0". Tapauksessa a) on tuoreimman säilyttäjän roolissa olevaa muuttuja, jota verrataan esittämällä muuttujan vieressä olevassa värillisessä palkissa sallitut arvot vihreällä ja ei-sallitut punaisella taustalla. Mikäli muuttujan arvo on sallitulla alueella, näytetään se muuttujasta lähtevällä ja palkin vihreälle alueelle osoittavalla vihreällä nuolella. Mikäli arvo ei ole sallittu, lähtee punainen nuoli muuttujasta palkin punaiselle alueelle. Tapauksessa b) askeltajan roolissa olevaa muuttujaa verrataan näyttämällä askeltajan roolikuvassa olevat sallitut arvot vihreinä ja ei-sallitut arvot punaisina. Mikäli

askeleet ovat seisahtuneet vihreän numeron kohdalle, on arvo sallittu. Erillistä palkkia sallituille ja ei-sallituille arvoille ei tarvita, koska jo askeltajan alustuksen jälkeen mahdolliset seuraavat ja edeltäneet arvot ovat tiedossa ja näkyvissä roolikuvassa. Mikäli vertailu olisi ollut muotoa "muuttuja > muuttuja2", olisi muuttuja2 molemmissa edellisistä tapauksista ilmestynyt näkyviin sen arvon kohdalle, joka sillä vertailuhetkellä oli.



Kuva 6: Vertailu erilaisilla rooleilla (Sajaniemi, 2002b).

2.6 Muuttujien roolit ohjelmoinnin oppimisen apuvälineenä

Sajaniemi & Kuittinen (in press) tutkivat empiirisellä kokeella parantaako muuttujien roolien opetus ohjelmoinnin alkeiden opetuksen yhteydessä oppimistulosta. Opetusta annettiin kolmella eri tavalla: perinteisellä tyylillä ilman muuttujien rooleja, muuttujien roolien avulla ilman niiden visualisointia ja muuttujien roolien avulla PlanAnilla tehtävän visualisoinnin kanssa. Kaikki ryhmät osallistuivat kurssin päätteeksi samaan kokeeseen, jonka tuloksia analysoitiin vertaamalla oppilaille syntyneitä mentaalisia malleja. Näin päädyttiin tekemään, koska tavallinen opettajien suorittama kokeiden arvostelu ei korreloinut oppilaiden todellisen ohjelmointitietämyksen kanssa.

Koska muuttujien roolit ovat ohjelmoinnin apuvälineitä, ne on parasta opettaa oppilaille ohjelmoinnin alkeiden opetuksen yhteydessä sitä mukaa, kuin ne esiintyvät malliohjelmissa (Sajaniemi & Kuittinen, in press). Muuttujien roolien ominaisuuksien opettamisen lisäksi oppilaille annetaan strategista tietoa roolien käytöstä ohjelmissa. Ensimmäisten ohjelmien tekeminen on aloittelijalle hyvin vaikeaa ja opettaja voi ohjata siinä kehoittamalla miettimään ohjelman tietotarpeita ja niiden myötä ohjelmassa mahdollisesti käytettäviä muuttujia ja niiden rooleja. Roolit käsitteinä helpottavat ohjelmoinnin periaatteiden selittämistä oppilaille.

Empiirisen kokeen tulokset osoittivat, että roolikoulutusta saaneet oppilaat olivat omaksuneet hyvin muuttujien roolit ja osasivat käyttää niitä myös uusissa tilanteissa. Oppilaista 35 prosenttia käytti rooleja koevastauksissaan, vaikka niitä ei kokeessa vaadittu. Tärkeä tulos oli myös, että roolipohjaista opetusta saaneet pystyivät ymmärtämään ja selittämään ohjelmia kokeneelle ohjelmoijalle tyypillisellä tavalla. Kumpikin rooliopetusta saanut ryhmä oli ohjelmointitietämykseltään perinteistä opetusta saanutta ryhmää parempi. Lisäksi ryhmä, jolle roolien käyttöä ohjelmissa animoitiin PlanAnin avulla, osoitti omaavansa syvempää ohjelmarakenteiden tuntemusta kuin ilman animointia koulutettu rooliryhmä ja perinteistä opetusta saanut ryhmä.

3 Metafora

MOT Sanakirjasto (2003) antaa sanan *metafora* selitykseksi kielikuvan tai vertauksen. Historiallisesti metaforia onkin käytetty kielikuvina lähinnä englantilaisessa kirjallisuudessa ja retoriikassa, mutta nykyisin metaforan merkitys on laajentunut (Hamilton, 1999). Lakoffin ja Johnsonin (1980) mukaan metaforan tarkoitus on selittää yksi asia toisen asian avulla.

Lakoffin (1993) mukaan metaforan avulla pystytään pääasiassa ymmärtämään abstrakteja käsitteitä ja järkeilemään abstrakteja asioita. Metafora on perusluonteeltaan käsitteellinen, ei kielellinen asia. Metaforinen kieli (metaphorical lanquage) on vain tapa tehdä metafora näkyväksi (surface manifestation). Metaforan avulla voidaan ymmärtää suhteellisen abstrakteja tai monimutkaisia asioita toisen konkreettisemman tai ainakin yksinkertaisemman asian avulla.

Esittelen tässä luvussa ensin metaforan toimintaperiaatteen esimerkin avulla sekä metaforan käyttötarkoituksen. Seuraavaksi kerron miten metaforaa käytetään tietojärjestelmissä ja annan siitä muutaman esimerkin. Lopuksi kuvaan menetelmän, jota voi käyttää metaforien suunnitteluun, toteutukseen ja arviointiin.

3.1 Metaforan toimintaperiaate ja käyttötarkoitus

Kuvaan seuraavaksi Glucksbergin ja Keysarin (1993) mukaisesti miten ”mies on sika” -metafora (kuva 7) ymmärretään ja mitä vaiheita sen ymmärtämiseen liittyy. Lause voidaan tulkita kahdella tavalla: kirjaimellisesti (linguistic) tai merkityksellisesti (speaker’s meaning). Kirjaimellinen tulkinta tehdään aina ensin, mutta varsinaisen merkityksen selvittämiseksi tarvitaan yleensä lisätietoja esimerkiksi asiayhteydestä (context).

Koska mies ei ole sika vaan ihminen, voidaan kirjaimellista tulkintaa pitää omituisena (”defective”). Tämän johdosta kuulija yrittää mahdollisesti tulkita asiaa metaforan avulla. Toisaalta metaforan ”ihminen ei ole saari” kirjaimellinen tulkinta on itsestäänselvä, jolloin ihminen alkaa myös etsiä sanojan tarkoittamaa merkitystä vaihtoehtoisella tavalla. Jos puhujan tarkoittamaa merkitystä yritetään tulkita metaforan avulla, lähdetään liikkeelle vertauksesta: ”mies on kuin sika”. Sian yleispiirteinä voidaan pitää ahneutta, lihavuutta ja likaisuutta, joten asiayhteydestä riipuen lauseen voi tulkita ainakin kolmella eri tavalla.

Kuva 7: ”Mies on sika”-metaforan käyttöä (Helsingin sanomat, 2003).

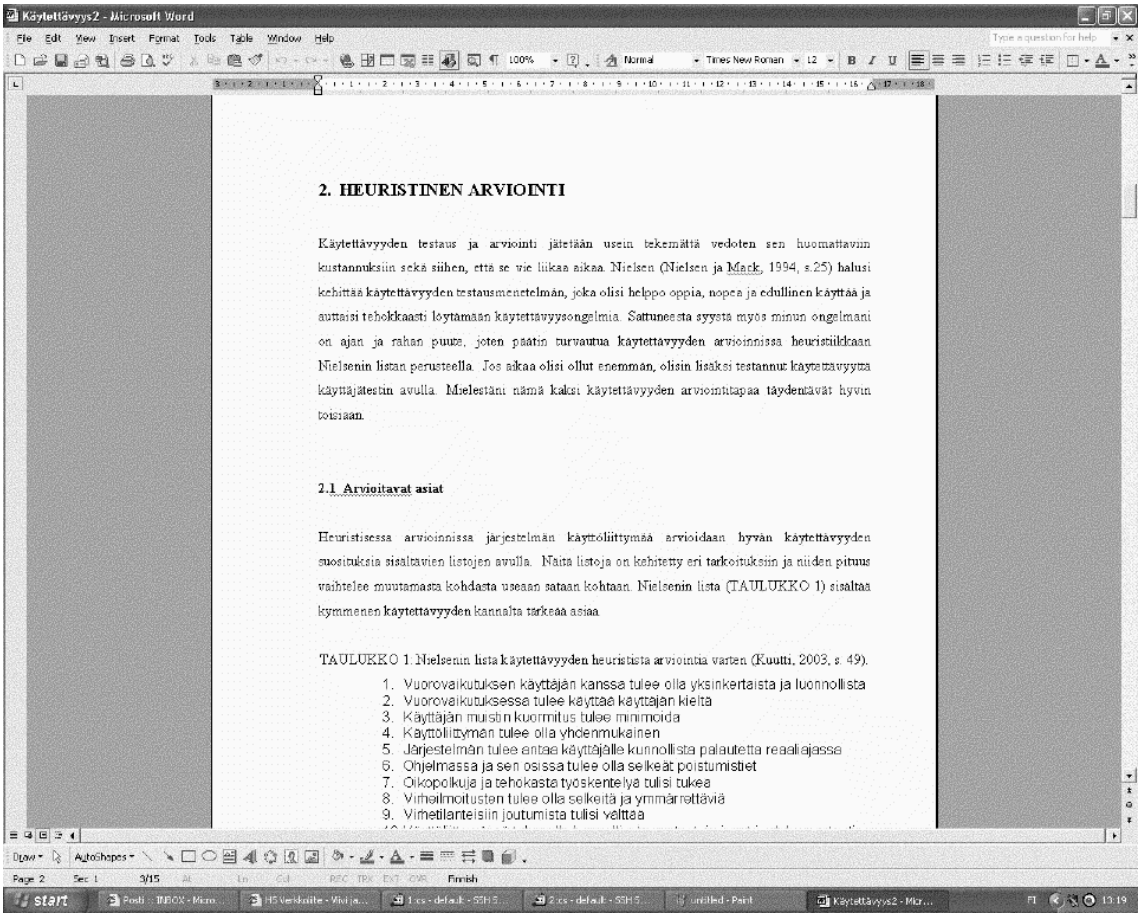
Metafora on usein myös harhaanjohtava, koska se on aina jossain määrin erilainen kuin sillä kuvattava asia: se voi salamannopeasti antaa ymmärtäjälleen oikean tai vääristyneen kuvan asiasta. Hamiltonin (2000) mukaan metaforinen vertaus on vain osittainen korostaen joitain erityisiä metaforan ja selitettävän asian yhteisiä ominaisuuksia. Esimerkiksi ”mies on sika”- metaforassa korostettava yhteinen ominaisuus voisi olla ahneus. Metaforinen vertaus sisältää myös metaforan ja selitettävän asian välisiä vähemmän korostuneita eroja. Tällainen ero voisi esimerkissäni olla, että sikaa syödään jouluna, miestä ei. Edellinen esimerkki on tietenkin metaforalle tyypilliseen tapaan riippuvainen myös kulttuurista, jossa metaforaa käytetään.

Metaforinen vertaus voi jäädä elämään, vaikka itse metaforaa esittävä asia olisi hävinnyt tai sen merkitys vähentynyt (Hamilton, 2000). Esmerkkinä tästä on kirjoituskonemetaforan käyttö tekstinkäsittelyohjelmissa (kuva 8), vaikka mekaaniset kirjoituskoneet ovat nykyisin hyvin harvinaisia.

Metaforan avulla yritetään konkretisoida olennaisia ominaisuuksia selitettävästä asiasta eli selitetään yksi asia käyttäen apuna jotain ennestään tuttua toista asiaa. Myös opetustehtävissä toimivat henkilöt käyttävät hyväkseen metaforaa, koska se helpottaa ihmisen oppimista. Carrol ja Mack (1999) ovat artikkelissaan jatkaneet metaforan käyttötarkoitusta vielä pidemmälle:

> *metaphors are kernel comparison statements whose primary function in learning is to stimulate active learner-initiated thought processes*

He tutkivat kuinka aloittelevat käyttäjät (noviisit) oppivat käyttämään tekstinkäsittelyohjelmaa. He havaitsivat, että käyttäjät mieluummin kokeilevat itse kuin lukevat ohjei-



Kuva 8: Kirjoituskone-metaforaa käytetään tekstinkäsittelyohjelmissa.

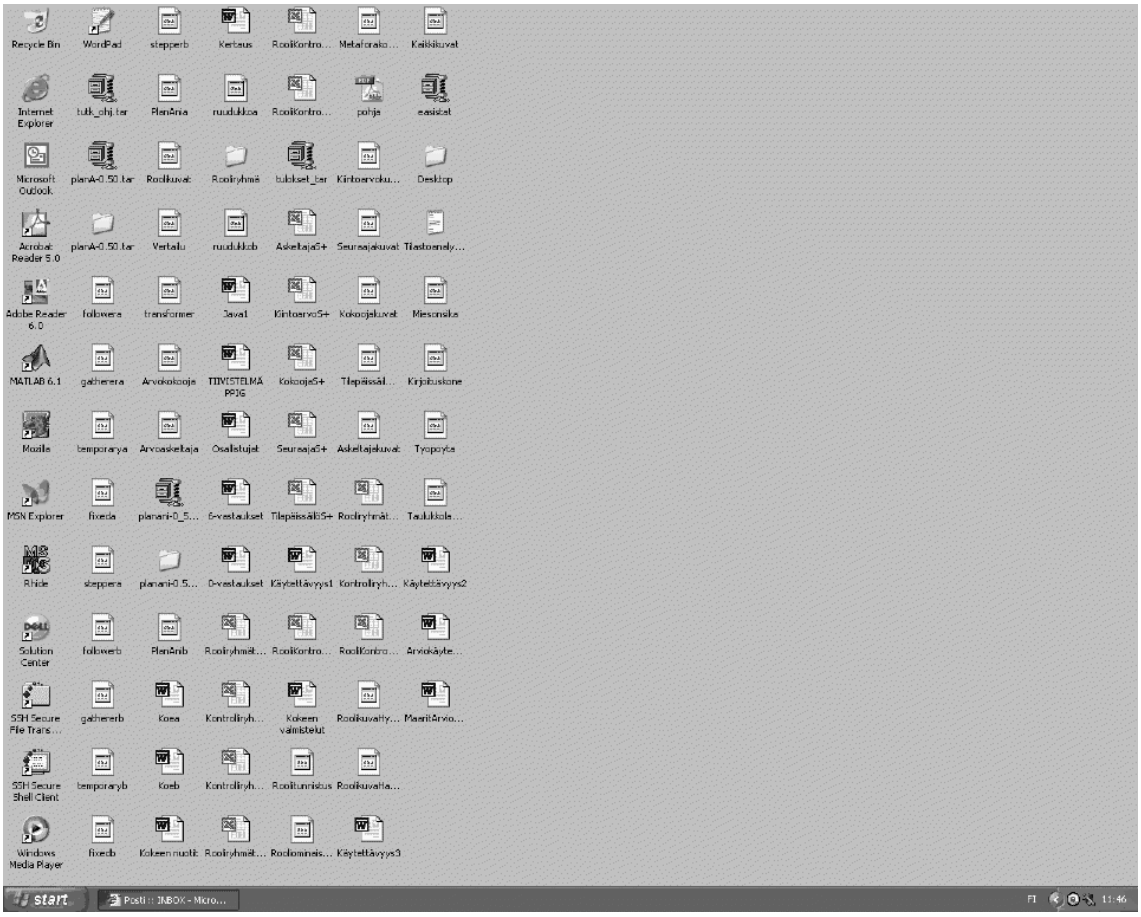
ta tai seuraavat järjestelmällistä askel-askeleelta opastusta. Vaikka käyttäjät olisivatkin lukeneet ohjeita, niiden tulkinta ja noudattaminen oli hankalaa. Ohjeet tuntuivat riittämättömiltä ratkaisemaan käyttäjien eteen tulevia monimutkaisia ongelmia. Sen sijaan he yrittivät itse kokeilemalla, päättelemällä ja järkeilemällä saada selville miten tekstinkäsittelyohjelma toimii ja miten sillä voisi eteen tulleita ongelmia ratkaista. Metaforien avulla käyttäjille voisi antaa vihjeitä tekstinkäsittelyohjelman toiminnasta ja siten aktivoida käyttäjän ajatusprosessia.

3.2 Metaforan käyttö tietojärjestelmissä

Kuvallisen metaforan käyttö tietojärjestelmissä tuli tarpeelliseksi tietokoneiden käytön yleistymisen myötä, kun siirryttiin komentopohjaisista käyttöliittymistä graafisiin. Tietokoneen sisältö yritetään visuaalisella esitystavalla tehdä käyttäjälle mahdollisimman havainnolliseksi ja helposti ymmärrettäväksi, jotta tämän ei enää tarvitse ponnistella niin paljoa oppiakseen käyttämään konettaan (Wozny, 1989). Yleisesti käytetty, Macintoshin tunnetuksi tekemä työpöytä-metafora (kuva 9) esittää tietokoneen toiminnot, ohjelmat, hakemistot ja tiedostot kuvallisina ikoneina, joita voidaan hiiren avulla suorittaa, käynnistää, katsella, siirrellä, poistaa jne.

Tietojärjestelmissä käytetään kahdentyyppisiä metaforia: sijaintia kuvaavia (physical) ja toiminnallisia (functional) (Wozny,1989). Esimerkki sijaintia kuvaavasta metaforasta on ikkunoiden käyttö. Käyttäjä voi käynnistää toimintoja eri ikkunoihin, muuttaa ikkunoiden järjestystä, paikkaa tai kokoa, siirtää tietoja ikkunasta toiseen jne. Sijaintia kuvaava metafora vastaa kysymykseen ”mitä on missä?”. Toiminnallisten metaforien tehtävänä on esittää tietojärjestelmien toimintoja käyttäjälle helposti ymmärrettävällä tavalla ja vastata kysymykseen ”miten se toimii?”. Esimerkkinä toiminnallisesta metaforasta on tekstin tai kohteen siirtäminen paikasta toiseen ”leikkaa ja liimaa” -toiminnon avulla. Tuttu askartelutermi on onnistuneesti siirretty tietokonemaailmaan aktivoimaan käyttäjiä yhdistelemään eri asioita keskenään ja säästämään aikaa hyödyntämällä aikaisemmin tehtyä työtä. Se saattaa myös houkutella käyttäjiä etsimään omatoimisesti tietoa verkosta oman työn tueksi ja näin motivoida hankkimaan uusia kokemuksia tietokoneesta.

Käyttäjien kannalta olisi myös tärkeää, että samat metaforat toistuvat useissa heidän käyttämissään sovelluksissa. Sekä Apple Computer että Microsoft ovatkin julkais-



Kuva 9: Työpöytä-metaforaa esittää tietokoneen toiminnot, ohjelmat, hakemistot ja tiedostot kuvallisina ikoneina, joita voi hiiren avulla esimerkiksi siirtää paikasta toiseen.

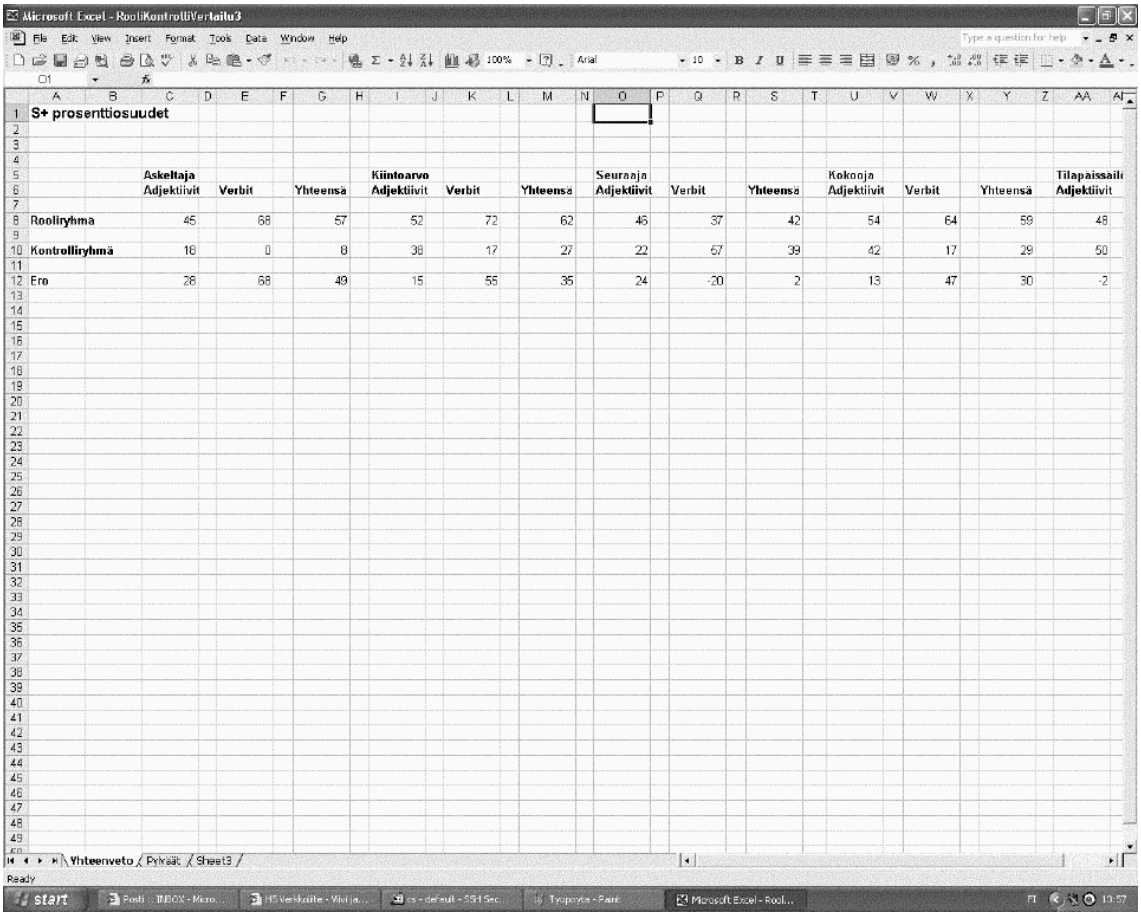
seet omat look-and-feel-standardinsa muiden ohjelmistotalojen käyttöön, jotta käyttäjät voisivat saada myös omien erikoisalojensa sovelluksia ennestään tutuilla käyttöliittymien ominaisuuksilla (Wozny, 1989).

Woznyn (1989) mukaan noviisit aloittaessaan tietojärjestelmän käytön toimivat aikaisempien kokemuksiensa pohjalta ja yrittävät käyttää järjestelmää kokeillen aiemmin omaksumiaan menetelmiä ja työtapoja, vaikka ne olisivat ”manuaalimaailmasta”. Jos he käyttävät tietokonetta harvoin, he eivät koskaan pääse tämän vaiheen yli, vaan käyttö aloitetaan aina tunnustelemalla ja kokeilemalla. Edistyneet käyttäjät etsivät uudesta järjestelmästä yhteneväisyyksiä (analogy) aikaisemmin käyttämiinsä järjestelmiin ja perustavat käyttönsä niille. Kaikille käyttäjäryhmille olisi parempi, jos tietojärjestelmien suunnittelussa olisi otettu huomioon mahdolliset käyttäjien ennestään tuntemat metaforat ja pyritty käyttämään niitä käyttäjälle tutulla tavalla. Jos sovellus poikkeaa metaforan toiminnasta, se pitäisi tehdä käyttäjälle selväksi ennen kuin hän itse huomaa sen ja mahdollisesti turhautuu. Lisäksi metaforien käytössä pitäisi suosia jatkuvuutta siten, että uudet sovellukset suunniteltaisiin tuntumaltaan ja mahdollisuuksien mukaan myös toiminnallisuudeltaan yhdenmukaisiksi muiden käytössä olevien sovellusten kanssa.

3.3 Esimerkkejä tietojärjestelmien metaforista

Ensimmäisiä metaforaa hyväksi käyttäviä sovelluksia oli taulukkolaskenta-ohjelma (kuva 10) Visicalc (Hamilton, 2000). Sen vahvuutena on ollut toiminnallisuuden nopea ymmärtäminen jo olemassa olevien käyttäjän taitojen perusteella. Laskentataulukko oli otettu käyttöliittymään suoraan käyttäjien ”manuaalimaailmasta”, sen idea ja käyttötapa olivat ennestään tuttuja ja sen käyttäminen oli helpompaa kuin paperisten taulukoiden. Käyttäjät pystyivät keskittymään työhönsä eli taulukon sisältöön eikä heidän tarvinnut sopeutua uuteen toimintatapaan. Tietokonejärjestelmä ikäänkuin hävisi käyttäjien näkyvistä - he vain tekivät työtään entistä paremmalla työkalulla.

Tekstinkäsittelijässä käytetty kirjoituskone-metafora (kuva 8) on myös ollut käyttäjille helposti omaksuttava samoista syistä kuin edellä esitetty taulukkolaskenta-ohjelma. Kirjoituskone-metafora on myös esimerkki siitä miten metafora voi vanhentua ja käydä vähemmän tärkeäksi. Tekstinkäsittelyn käyttäjistä vain iäkkäimmät ovat joskus kirjoittaneet mekaanisella kirjoituskoneella tai nähneet sellaisen (Hamilton, 2000).



Kuva 10: Laskentataulukko-metaforaa käyttää myös nykyään esimerkiksi Excel-taulukkolaskentaohjelmisto.

Roskakori-metaforaa on pidetty erittäin kuvaavana, selkeänä ja toiminnaltaan nopeasti ymmärrettävänä, mutta Macintosh-ympäristössä toteutettuna siihen on liittynyt myös hämmennystä aiheuttava ominaisuus: kun levykeaseman siirtää roskakoriin, levyke ponnahtaa ulos asemasta. Koska roskakori on ymmärretty tiedostojen tuhoamispaikkana tai niiden väliaikaisena säilytyspaikkana ennen lopullista tuhoamista, kuvitellaan että myös levykkeen sisältö tuhoutuu roskakoriin vietäessä. Näin ei kuitenkaan ole, vaan levykkeen sisältö pysyy levykkeellä eikä siirry roskakoriin tuhottavaksi. Tällaiset metaforan ”väärinkäytöt” voivat ärsyttää ja turhauttaa käyttäjiä ja aiheuttaa heille turhaa kognitiivista kuormitusta (cognitive dissonance).

3.4 Menetelmä metaforien valintaan

Hyvin valitulla metaforalla voidaan helpottaa uuden tietojärjestelmän käytön aloittamista ja aktivoida ja innostaa käyttäjää etsimään ja kokeilemaan järjestelmän ominaisuuksia, mikä jo sinänsä parantaa järjestelmän käytettävyyttä (Alty *& al.*, 2000). Mikäli metafora on huonosti valittu, se saattaa johtaa käyttäjäänsä harhaan, rajoittaa tämän ajattelua ja nostaa kynnystä käyttää järjestelmää.

Käytännön ohjeita metaforan valintaan löytyy kirjallisuudesta vain vähän ehkä juuri siksi, että yleispätevien ohjeiden antaminen on hyvin vaikeaa. Alty *& al.* (2000) on tietoliikennealan järjestelmiin suunniteltujen metaforien käytännön kokemuksiin nojaten kehittänyt yhden ohjeiston metaforien valintaan, toteutukseen ja arviointiin. Ohjeisto kehitettiin EU-komission RACE-osaston käynnistämän MITS-projektin (Metaphors for Integrated Telecommunications Services) tuloksena vuosien 1993-1997 aikana. Ohjeisto viimeisteltiin työryhmissä, jotka koostuivat useista maista tulleista ohjelmistotuotannon ammattilaisista. Ammattilaisten mielipide oli tärkein kriteeri valittaessa lopulliseen ohjeistoon tulevia menetelmiä ja työkaluja. Ohjeisto koostuu kuudesta osasta: toiminnallinen määrittely, metaforien suunnittelu, metaforien analysointi, metaforien toteutus, metaforien testaus ja palautteen kerääminen. Koska tutkimukseni kohdistuu jo valittujen metaforien analysointiin, testaukseen ja palautteen keräämiseen, painotan esityksessäni eniten niitä.

Toiminnallinen määrittely: Järjestelmän toiminnallisen määrittelyn tuloksena tiedetään mitä toimintoja tulevan järjestelmän täytyy sisältää, mikä luo pohjan metaforien valinnalle. Tulevan järjestelmän piirteiden kuvaus on edellytys metaforien ja järjestel-

män välisten yhteneväisyyksien ja erojen analysoinnille. Järjestelmän tulevia käyttäjiä voidaan pyytää kuvaamaan uuden järjestelmän toiminnallisuutta, jolloin he saattavat tehdä sen heille ennestään tuttujen metaforien avulla. Tässä vaiheessa on kuitenkin keskityttävä nimenomaan järjestelmän toiminallisuuteen eikä vielä metaforiin ja niiden ominaisiin.

Metaforien suunnittelu: Metaforien suunnitteluvaiheessa yritetään löytää järjestelmän toiminnallisuutta kuvaavia metaforia. Siinä tulee huomioida käyttäjien nykyinen osaaminen ja muiden heidän käytössään olevien järjestelmien käyttämät metaforat. Parhaiten se onnistuu tutustumalla käyttäjien maailmaan kuten työpisteisiin, työskentelytapoihin, käytettyihin termeihin ja tietenkin sovellusalueeseen käyttäjän näkökulmasta. Metaforien täytyy olla nimenomaan käyttäjilleen kuvaavia, joten tässä vaiheessa kannattaa olla luova ja keksiä useita vaihtoehtoisia ratkaisuja.

Metaforien analysointi: Tässä vaiheessa analysoidaan metaforan (M) ja järjestelmän (S) ominaisuudet. Ne voidaan jakaa metaforasuunnittelun kannalta oleellisiin kolmeen luokkaan. *S+M+* -luokkaan kuuluvat sekä järjestelmästä että metaforasta löytyvät ominaisuudet. *S+M-* -luokkaan kuuluvat ominaisuudet, jotka kuvaavat järjestelmää, mutta niitä ei löydy metaforasta. *S-M+* -luokkaan kuuluvat ominaisuudet, jotka löytyvät metaforasta, mutta eivät järjestelmästä.

Mikäli S-M+ ominaisuuksien osuus on suuri verrattuna S+M+ ominaisuuksien määrään, saattaa jatkuva puuttuviin ominaisuuksiin törmääminen ärsyttää käyttäjää ja aiheuttaa tälle turhaa kuormitusta (conceptual baggage) (Alty *& al.*, 2000). On parempi antaa käyttäjälle mahdollisuus "löytöretkeilijän" iloihin ja valita metafora, jossa S+M- ominaisuuksien määrä olisi suuri. Tämä myös aktivoi käyttäjää etsimään uusia ominaisuuksia.

Metaforien toteutus: Metaforan toteutuksessa täytyy ottaa huomioon sen esitystapa, realistisuus, johdonmukaisuus ja sopivuus. On tärkeää, että käyttäjä tunnistaa metaforan välittömästi sen nähtyään, koska hän tekee sen perusteella oletuksia järjestelmän toiminnasta. Mikäli metaforan ulkoasu on tarkka kopio reaalimaailmasta, käyttäjä olettaa toiminnallisuudenkin olevan sama kuin reaalimaailmassa. Tämä toisaalta rajoittaa käyttäjän ajattelua ja toisaalta saattaa myös turhauttaa, mikäli järjestelmä ei sisälläkään ihan kaikkea samaa toiminnallisuutta. Tietojärjestelmässä käytetään yleensä useita metaforia samanaikaisesti ja käyttäjää helpottaa, jos metaforat on valittu johdonmukaises-

ti. Esimerkiksi ”työpöytä”-metaforan yhteydessä käytettyjä muita metaforia ovat työpöydällä mahdollisesti olevat kansiot, laskin, roskakori jne. On myös metaforia, jotka ovat käytössä useissa järjestelmissä, esim. leikkaa-liimaa-toiminto kohteen siirtelyyn tiedoston sisällä tai tiedostojen välillä. Nämä metaforat pitäisi esittää yhdenmukaisina kaikissa järjestelmissä, jotta ne eivät aiheuttaisi sekaannuksia käyttäjille. Metafora ei saa olla rasistinen, seksistinen, ikäsyrjintää harrastava eikä muillakaan tavoilla aiheuttaa mielipahaa käyttäjilleen. Ohjelmistoja tekevän yrityksen olisi parempi metaforia suunnitellessaan ottaa huomioon myös yrityskuva, jonka he haluavat itsestään järjestelmän käyttäjille antaa.

Metaforien testaus: Metaforien testauksen tarkoitus on arvioida metaforien sopivuutta niiden käyttöympäristössä ja näin saada palautetta niiden kehittämistä varten. On tärkeää testata metaforat käyttäjien avulla käyttäjille tutussa ympäristössä mieluiten käyttäjien varsinaisilla työtehtävillä. Alty *& al.* (2000) pitää järkevänä videonauhoituksen tekemistä testaustilanteessa. Sen perusteella voidaan nähdä käyttäjäkohtaisia eroja sekä saadaan aineistoa järjestelmän suunnittelijoita varten. Lisäksi käyttäjiltä pitäisi saada verbaalista palautetta järjestelmän metaforista.

Testauksen aikana tulisi arvioida ymmärtävätkö käyttäjät metaforan siten kuin oli tarkoitus vai tekevätkö käyttäjät metaforan takia vääriä oletuksia. Ovatko metaforat tarpeeksi realistisia, jotta käyttäjät tunnistavat ne vai liiankin realistisia rajoittaen näin liikaa käyttäjän mielikuvitusta. Lisäksi täytyy arvioida pystyykö järjestelmää laajentamaan käyttäen samoja metaforia ja ovatko metaforat johdonmukaisesti valittuja ja tulevatko ne esimerkiksi ”samasta perheestä”. Täytyy myös varmistua siitä, että metaforat ovat järjestelmän käyttöympäristöön sopivia ja luonnollisia.

Palautteen kerääminen: Palauteen kerääminen järjestelmän käyttäjiltä auttaa järjestelmän jatkokehityksessä ja seuraavia metaforia suunnitellessa. Esimerkiksi käyttäjiltä voi tulla hyviä ehdotuksia järjestelmään otettavista uusista ominaisuuksista, joita he ovat käytetyn metaforan takia yrittäneet järjestelmästä löytää. Tai käyttäjiltä voi saada palautetta epäloogisesti toimivasta metaforasta.

4 Empiirinen koe

PlanAnissa käytettävät roolikuvat voidaan käsittää kuvallisiksi metaforiksi. Metafora-tutkimuksen mukaan oikein valittu metafora helpottaa asian ymmärtämistä ja stimuloi ajatusprosesseja (Carrol & Mack, 1999). Sajaniemen & Kuittisen (in press) mukaan PlanAnilla toteutettu ohjelman visualisointi syventää oppilaiden ohjelmatuntemusta. Edellisen perusteella voisi kuvitella, että PlanAnin metaforat ovat oikein valittuja ja siten auttavat oppilaita omaksumaan ohjelman sisältöä syvemmällä tasolla kuin ilman visualisointia toteutettu roolipohjainen opetus.

Halusin kokeella todistaa, että muuttujien rooleista käytetyt metaforat välittävät oppilaille roolien ominaisuuksia ja näin helpottavat rooliteorian omaksumista. Koetta varten tehtiin PlanAnista myös toinen versio erilaisilla roolikuvilla. Vaihtoehtoiset lumekuvat valitsin siten, että ne ominaisuuksiltaan mahdollisimman vähän kuvasivat muuttujien rooleja. Mikäli metaforalla olisi muuttujien roolien visualisoinnissa jotain merkitystä, tulisi kokeen osoittaa PlanAnin alkuperäisillä kuvilla parempaa tulosta kuin lumekuvilla.

Tämä koe voidaan mielestäni ajatella osana Alty *& al:*in (2000) metaforien valintaan, toteutukseen ja arviointiin kehittämää menetelmää. Menetelmä koostuu kuudesta osasta: toiminnallinen määrittely, metaforien suunnittelu, metaforien analysointi, metaforien toteutus, metaforien testaus sekä palautteen keräys ja analysointi. Ideana on kuvata järjestelmän toimintaa ja sen ominaisuuksia ja verrata niitä ehdolla olevien, järjestelmää kuvaavien metaforien ominaisuuksiin.

PlanAnissa metaforia on käytetty kuvaamaan animoitavan ohjelman muuttujia ja niiden roolien ominaisuuksia. Metaforat kuvaavat siis PlanAnin käsittelemiä tietoja, eivät PlanAni-järjestelmää. Tästä syystä olen soveltanut menetelmää korvaamalla järjestelmän toiminnallisen määrittelyn muuttujien roolien määrittelyllä.

Alty *& al:in* (2000) suunnitteleman ohjeiston tarkoituksena on verrata useita vaihtoehtoisia metaforia järjestelmän ominaisuuksiin ja valita toteutettavaksi niistä paras vaihtoehto. Tässä kokeessa PlanAnista oli toteutettuna kaksi erilaisilla roolikuvilla varustettua versiota, joita oli tarkoitus vertailla käyttäjien antaman palautteen avulla.

4.1 Koesuunnitelma

Kokeessa käytettiin koehenkilöinä Sajaniemen & Kuittisen (2003b) koeryhmää, joka oli saanut ohjelmoinnin alkeiden opetusta muuttujien roolikäsitteen avulla, mutta ei ollut käyttänyt PlanAnia. Koehenkilöt olivat nähneet oikeat PlanAnissa käytetyt roolikuvat mustavalkoisina paperisessa, opetukseen liittyvässä materiaalissa noin puolitoista vuotta ennen kokeen suorittamista. Lumekuvia koehenkilöt eivät olleet aikaisemmin nähneet.

Rooliteoria palautettiin koehenkilöiden mieleen kirjallisella kertausmateriaalilla siten, ettei roolikuvia missään vaiheessa esitelty koehenkilöille. Koehenkilöt testattiin kertauksen jälkeen tasokokeella, jossa piti tunnistaa muuttujien rooleja. Kokeeseen osallistuvat jaettiin tasokokeen perusteella kahteen taidoiltaan yhtä vahvaan ryhmään, jotka saivat testattavakseen erilaiset kuvat. Alkuperäiset roolikuvat sai testattavakseen *rooliryhmä* ja lumekuvat annettiin *kontrolliryhmälle*. Kokeessa käytettiin between-subject designia, millä vältettiin oppimisefekti kuvien välillä.

Koe koostui useasta eri osiosta, joilla pyrittiin kartoittamaan kuvien ominaisuuksia, testaamaan niiden mieleen jäämistä, saamaan palautetta niiden tarpeellisuudesta ja osuvuudesta sekä mittaamaan roolien tunnistamistaidon kehittymistä kokeen aikana. *Ensimmäinen osio* pyrki kartoittamaan roolikuvien herättämiä mielikuvia, joita oli tarkoitus verrata taulukossa 1 lueteltuihin roolien ominaisuuksiin. *Kokeen toinen osio* pyrki testaamaan roolikuvia käytännössä eli koehenkilöt tutustuivat ensimmäisessä osiossa arvioimiinsa roolikuviin seuraamalla yksinkertaisen ohjelman suoritusta PlanAnilla. Tämä kokeen osa suoritettiin, koska Alty *& al:in* (2000) mukaan on tärkeää arvioida käytettyjä metaforia siinä yhteydessä ja ympäristössä kuin niitä tullaan käyttämään. PlanAni animoi muuttujien rooleja metaforien avulla, joten arviointi oli syytä tehdä sitä käyttäen. Alty *& al.* suositteli myös testauksen nauhoittamista videolle, mikä testattaessa järjestelmän metaforia mielestäni onkin perusteltua. Koska tässä testattiin pelkästään sitä, miten metafora kuvaa muuttujan roolia eikä PlanAnin toimintaa tai sen käyttöä, en pitänyt tarpeellisena nauhoittaa koetilannetta videolle. *Kokeen kolmas osio* pyrki selvittämään jäivätkö PlanAnissa käytetyt roolikuvat koehenkilöiden mieleen sekä keräämään palautetta niiden osuvuudesta ja tarpeellisuudesta. *Neljännessä osiossa* roolihenkilöiden piti kokeen kertausosiossa tehdyn tasotestin tapaan tunnistaa ohjelmakoodista muuttujien rooleja. Tarkoituksena oli testata oliko koehenkilöiden roolituntemus kehittynyt kokeen aikana.

4.1.1 Koehenkilöt

Sajaniemi & Kuittinen (2003b) ovat tutkineet parantaako muuttujien roolien opetus ohjelmoinnin alkeiden opetuksen yhteydessä oppimistulosta. Ohjelmoinnin alkeita opetettiin kolmella eri tavalla: perinteisellä tyylillä ilman muuttujien roolikäsitettä, muuttujien roolien avulla ilman niiden visualisointia ja muuttujien roolien avulla PlanAnilla tehtävän visualisoinnin kanssa. Koehenkilöt tähän kokeeseen valittiin siten, että he olivat kuuluneet Sajaniemen & Kuittisen (2003b) rooliopetusta ilman visualisointia saaneeseen koeryhmään.

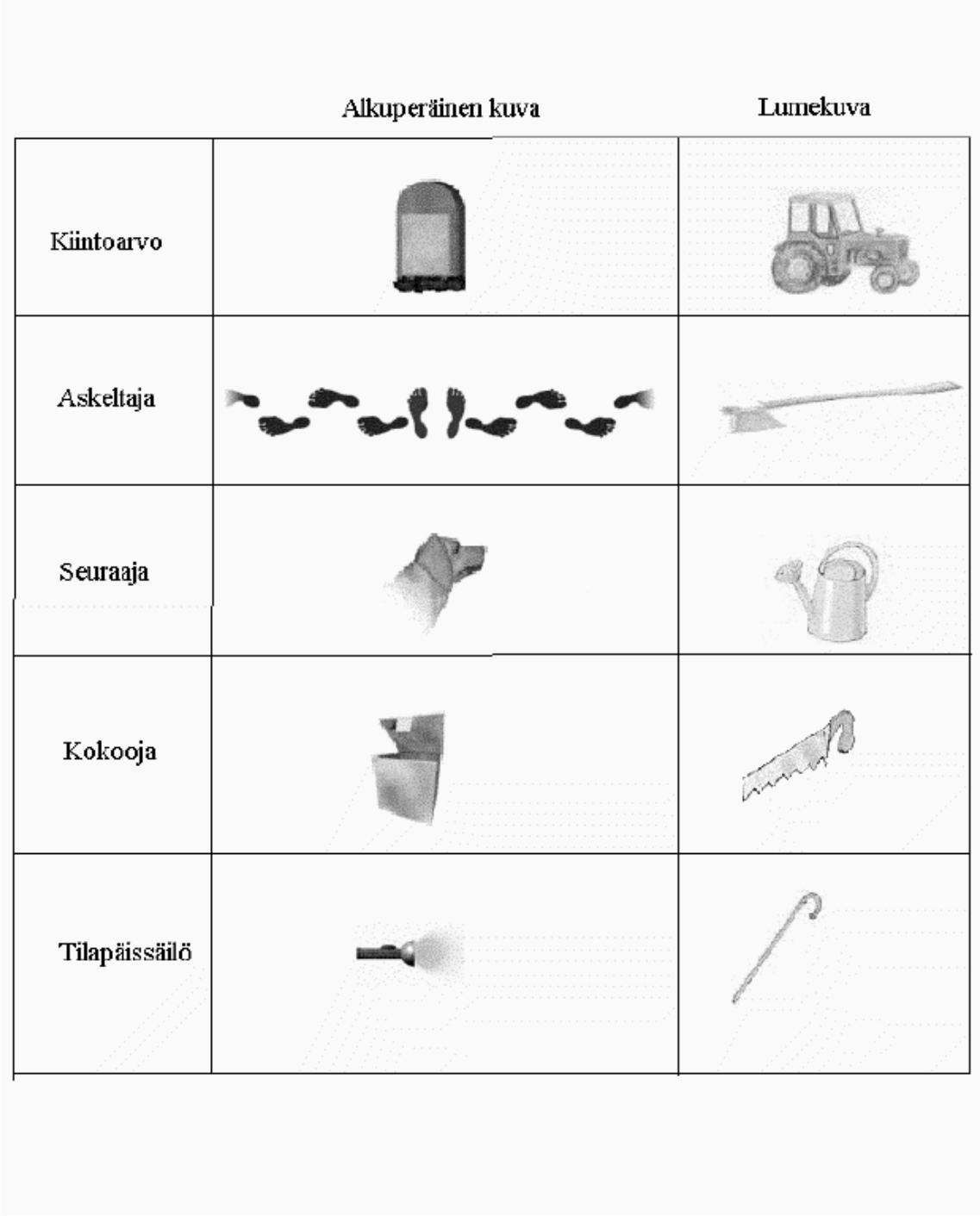
Kutsu kokeeseen lähetettiin kaikille yllämainittuun ryhmään kuuluneille ja heistä 13 suostui osallistumaan. Koehenkilöistä 12 oli miespuolisia ja yksi oli nainen. Jako tasavahvoihin ryhmiin tehtiin roolien kertausvaiheen jälkeen pidetyn tasotestin perusteella: 7 rooliryhmään ja 6 kontrolliryhmään. Osallistujille annettiin palkkioksi lounasliput paikalliseen lounasravintolaan.

4.1.2 Materiaalit

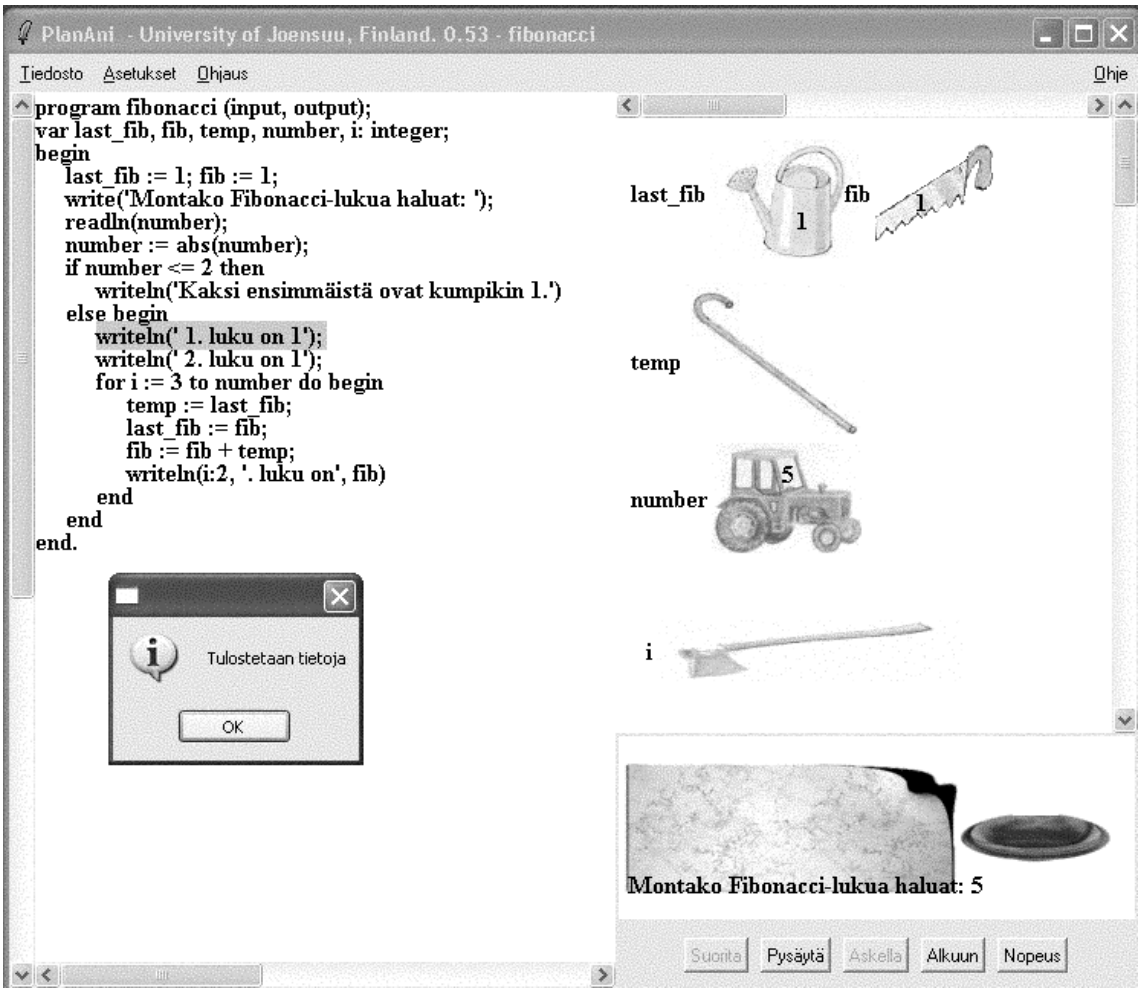
Rooliteorian kertausta varten molemmat koeryhmät saivat saman kirjallisen kertausmateriaalin (liite 1; Sajaniemi, 2004). Tasokokeessa oli kolme pientä Pascal-kielistä ohjelmaa, joista täytyi tunnistaa muuttujien roolit (liite 2; Sajaniemi, 2004).

Ensimmäistä varsinaista koeosiota varten oli erikseen kysymyspaperit rooliryhmälle (liite 3) ja kontrolliryhmälle (liite 4). Ensimmäisenä tehtävänä molemmilla ryhmillä oli kirjoittaa neljä verbiä annetuista kuvista (kuva 11). Toisena tehtävänä oli kirjoittaa neljä adjektiivia annetuista kuvista. Kuvat esittivät rooleja kiintoarvo, askeltaja, seuraaja, kokooja ja tilapäissäilö.

Toista koeosiota varten PlanAnista oli rooliryhmän käyttämän alkuperäisen version (kuva 2) lisäksi lumekuvilla (kuva 12) tehty kontrolliryhmälle annettu versio. Animoitava ohjelma oli molemmissa sama Pascal-kielinen Fibonacci-ohjelma. PlanAnista oli käytössä suomenkielinen versio, jotta muuttujien roolien nimet vastaisivat kertausmateriaalissa olleita roolinimiä. Lisäksi PlanAni täytti koko näyttöruudun eikä mitään muita ohjelmia päässyt kokeen aikana käyttämään. Tehtävänä (liite 5) molemmilla koeryhmillä oli antaa ohjelmalle syötteeksi luku 6 ja kirjoittaa vastauspaperiin ohjelman suorituksen aikana muuttujille sijoitetut arvot.



Kuva 11: Kokeessa eri rooleista käytetyt alkuperäiset ja lumekuvat.



Kuva 12: Ohjelman visualisointia PlanAnilla lumekuvien avulla.

Kolmannessa koeosiossa haluttiin palautetta PlanAnilla suoritettuun osioon. Koska PlanAnista oli kaksi versiota, myös palautekyselystä oli omat versiot rooliryhmälle (liite 6) ja kontrolliryhmälle (liite 7). Tehtävässä 4 oli esillä PlanAnin roolikuvat ja jokaisesta kuvasta piti kertoa mitä roolia se esittää. Lisäksi piti arvioida kouluarvosanalla (4-10) kuinka hyvin kuva roolia esittää ja kirjoittaa tai piirtää kuva, joka esittäisi roolia kympin arvoisesti. Molemmat ryhmät jatkoivat arviointia tehtävässä 5 (liite 8) ottamalla kantaa siihen miten hyödyllistä roolien kuvilla esittäminen on, onko siitä jotain apua tai voisiko siitä olla jotain haittaa.

Viimeisenä osiona oli molemmille koeryhmille yhteinen tehtävä 6 (liite 9; Sajaniemi, 2004), jossa täytyi tunnistaa muuttujien roolit kolmesta pienestä Pascal-kielisestä ohjelmasta. Ohjelmat olivat erilaisia kuin kertauksen päätteeksi pidetyssä tasokokeessa käytetyt ohjelmat.

Kokeen paperiset materiaalit oli niitattu yhteen kolmeksi erilaiseksi nipuksi, joista jokaisen ensimmäisenä sivuna oli kansilehti sisältäen päivämäärä- ja nimitiedot. Kertausmateriaalinippu tasokokeineen oli yhteinen molemmille koeryhmille, rooliryhmä sai tehtävänipun oikeilla roolikuvilla ja kontrolliryhmä lumekuvilla.

Lisäksi kokeenjohtajia varten oli tehty ohjeet (liite 10) kokeen toistamiseksi mahdollisimman samanlaisena. Ohjeet sisälsivät kokeenjohtajien vuorosanat, tekemiset sekä koetehtävien kestoajat.

Koetta varten oli varattu kaksi mikroluokkaa, joista toiseen oli asennettu rooliryhmää varten PlanAnin oikeat versiot ja toiseen kontrolliryhmää varten tehdyt, lumekuvilla varustetut PlanAnin versiot.

4.1.3 Koejärjestely

Muuttujien roolien kertaus Koehenkilöille annettiin kertausmateriaali ja tasokoe (liite 1 ja liite 2; Sajaniemi,2004) nipussa, jonka ensimmäisenä sivuna oli päivämäärä- ja nimitiedot keräävä kansilehti. Koehenkilöitä pyydettiin kirjoittamaan nimensä kansilehdelle, perehtymään kertausmateriaaliin ja tekemään materiaalin lopussa olevat tehtävät. Tehtävien tekemisessä voi vapaasti käyttää hyväksi kertausmateriaalia. Aikaa tehtävien tekemiseen oli 15 minuuttia.

Tasokokeen suorittamisen jälkeen koehenkilöt saivat poistua tauolle, jonka aikana kokeenjohtajat arvostelivat tasokokeen ja jakoivat osallistujat sen perusteella kahteen yhtä tasokkaaseen ryhmään.

Roolikuvien ominaisuuksien kartoitus Tässä vaiheessa koehenkilöt oli jaettu rooliryhmään, joka sai arvioitavakseen alkuperäiset PlanAnin kuvat ja kontrolliryhmään, joka sai arvioitavakseen lumekuvat. Ryhmät suorittivat kokeen eri tiloissa. Koehenkilöille kerrottiin, että tehtävät tehdään siinä järjestyksessä ja aikataulussa kuin ohjaaja sanoo ja että sivua saa kääntää vain silloin kun ohjaaja sanoo. Kokeen kannalta oli erittäin tärkeää, että myöhempänä olevia tehtäviä ei saa etukäteen kurkkia eikä aikaisemmin tehtyjä tehtäviä jälkikäteen muuttaa.

Kokeen ensimmäisenä varsinaisena tehtävänä piti kirjoittaa kustakin roolikuvasta neljä adjektiivia. Koehenkilöt saivat paperilla testattavat roolikuvat (liite 3 ja liite 4), joita he eivät siihen mennessä olleet nähneet yhdistettyinä rooleihin. Aikaa tehtävän tekemiseen oli viisi minuuttia, jonka jälkeen ohjaaja pyysi kääntämään esiin seuraavan tehtävänipun tehtävän.

Kokeen toisena tehtävänä piti kirjoittaa kustakin roolikuvasta neljä verbiä. Aikaa tehtävän tekemiseen oli viisi minuuttia, jonka jälkeen ohjaaja pyysi taas kääntämään seuraavan sivun esille.

Roolikuvien toiminta käytännössä Kokeen toinen osio aloitettiin tietokoneella PlanAni-sovelluksella. Tietokoneisiin oli jo ennen koetta käynnistetty PlanAnit, jonka jälkeen näytöistä oli sammutettu virta. Tämän tehtävän alussa ohjaaja pyysi koehenkilöitä laittamaan näyttöihin virran päälle. Koehenkilön tehtävänä oli tutustua Fibonacci-ohjelmaan ja muuttujien roolikuviin PlanAnia käyttäen ja samalla vastata paperilla jaettuihin kysymyksiin (liite 5). Aikaa oli 10 minuuttia, jonka jälkeen ohjaaja pyysi laittamaan näytöistä virran pois ja kääntämään tehtävänipun seuraavan tehtävän esiin.

Roolikuvien muistaminen ja palautteen keruu Kokeen kolmannessa osiossa testattiin miten hyvin koehenkilöiden mieleen olivat jääneet juuri PlanAnissa käytetyt kuvat ja niitä vastaavat roolit. Tehtävässä 4 (liite 6 ja liite 7) koehenkilöitä pyydettiin kertomaan mitä roolia kukin PlanAnissa käytetty roolikuva esittää. Sajaniemen & Kuittisen (2003) sanallisesti ilmaisemat roolikuvaukset olivat esillä tehtäväpaperin alalaidassa. Roolikuvauksissa olivat edustettuina kaikki taulukossa 2 listatut vaihtoehdot.

Taulukko 2: Muuttujien roolien vapaamuotoinen määrittely.

Rooli	*Määrittely*
Kiintoarvo	Muuttuja, jonka arvo ei alustuksen jälkeen muutu.
Askeltaja	Muuttuja, joka saa kaikki arvonsa alkuarvon asetuksen jälkeen ennustettavissa olevasta sarjasta.
Tuoreimman säilyttäjä	Muuttuja, joka säilyttää viimeisimmän arvon jostakin käsiteltävänä olevasta sarjasta arvoja.
Sopivimman säilyttäjä	Muuttuja, joka säilyttää parhaan tai sopivimman arvon jostakin käsiteltävänä olevasta sarjasta arvoja.
Kokooja	Muuttuja, saa arvonsa summaamalla tai muulla tavalla kumuloimalla käsiteltävänä olevasta sarjasta arvoja.
Seuraaja	Muuttuja, joka saa arvokseen jonkin toisen muuttujan vanhan arvon.
Yksisuuntainen lippu	Muuttuja, jolla on vain kaksi mahdollista arvoa ja joka ei voi saada enää alkuarvoaan muutoksen jälkeen.
Järjestelijä	Taulukko, jota käytetään alustuksen jälkeen vain tietojen järjestämiseen toisella tavalla.
Tilapäissäilö	Muuttuja, joka säilyttää arvonsa vain hyvin vähän aikaa.
Muu	Mikä tahansa muu muuttuja.

Lisäksi koehenkilöiden piti arvioida kouluarvosanalla (4-10) kuinka hyvin kuva esittää roolia sekä ehdottaa heidän mielestään sopivampaa kuvaa roolille. Aikaa tehtävän tekemiseen oli kymmenen minuuttia, jonka jälkeen ohjaaja pyysi koehenkilöitä kääntämään seuraavan tehtävänipun sivun esiin.

Tehtävässä 5 (liite 8) koehenkilöt ottivat kantaa siihen miten hyödyllistä roolien esittäminen kuvalla on, voisiko siitä olla apua tai peräti haittaa. Aikaa oli 10 minuuttia, jonka jälkeen ohjaaja pyysi koehenkilöitä kääntämään esiin tehtävänipun seuraavan sivun.

Roolien tunnistaminen ohjelmista Kokeen viimeisessä osiossa koehenkilöt saivat eteensä kolme pientä ohjelmaa (liite 9; Sajaniemi, 2004), joissa käytettyjen muuttujien roolit heidän piti tunnistaa. Aikaa oli käytettävissä 10 minuuttia, jonka jälkeen ohjaaja keräsi vastauspaperit pois kiittäen samalla kokeeseen osallistujia.

4.1.4 Esikoe

Testasin koejärjestelyt esikokeen avulla ennen varsinaista koetta. Koehenkilöinä oli kaksi tietojenkäsittelytieteen ammattilaista, joilla oli ohjelmointikokemusta muutamalta vuodelta. He eivät entuudestaan tunteneet muuttujien rooliteoriaa, mutta opiskelivat sen kokeeseen kuuluvan kertausmateriaalin avulla. Ensimmäinen koehenkilöistä teki kokeen kontrolliryhmän materiaaleilla, jonka jälkeen korjasin materiaalien ja koejärjestelyjen puutteita. Toinen koehenkilöistä teki esikokeen korjatuilla rooliryhmän materiaaleilla, jonka tulosten perusteella kehitin materiaaleja ja koejärjestelyä edelleen.

Esikokeiden tulosten perusteella tarkensin kokeenjohtajien ja koetehtävien ohjeita sekä arvioin koetehtävien suoritusaikoja. Lyhensin adjektiivi- ja substantiivi-tehtäviä, koska en halunnut kokeen venyvän kaksituntiseksi. Kokeen viimeisen tehtävän vaihdoin kummankin esikokeen jälkeen. Alkuperäisessä koeversiossa viimeisessä tehtävässä piti kirjoittaa kokeessa esiteltyjen viiden eri roolin ominaisuuksia. Tehtävän tarkoituksena oli testata koehenkilöiden roolien omaksumista. Ensimmäinen testikoehenkilö kirjoitti lähes täydelliset roolikuvaukset jokaisesta roolista, joten tehtävä törmäsi heti kattoefektiin. Seuraavaan koeversioon laitoin kokeen viimeiseksi tehtäväksi saman roolien tunnistustehtävän kuin oli ollut kokeen alussa kertaustehtävänä, mutta nyt roolit piti tunnistaa ilman kertausmateriaalia. Koska koehenkilö vastasi tehtävään muistelemalla sitä, mitä oli vastannut aikaisemmalla kerralla, päätin vaihtaa lopulliseen koeversioon kertausmateriaalista poikkeavat ohjelmat.

4.2 Tulokset

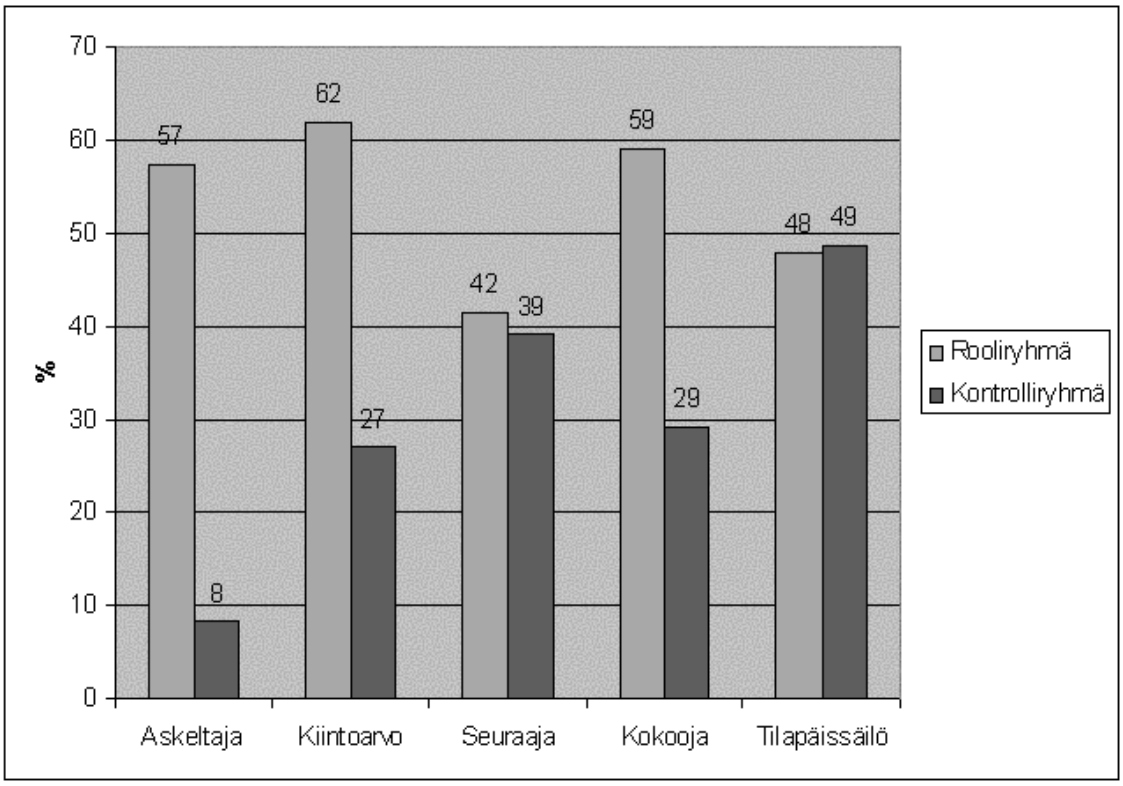
Kokeen tuloksia tarkastellaan laskennallisesti neljästä näkökulmasta: roolikuvien roolia heijastavat ominaisuudet, roolien tunnistaminen kuvan perusteella, roolikuvien käytön tarpeellisuus ja roolien tunnistaminen ohjelmasta.

4.2.1 Metaforien ominaisuudet

Koehenkilöitä oli pyydetty luettelemaan rooleja esittäville kuville adjektiiveja ja verbejä. Tällä tehtävällä halusin saada selville miten hyvin kuvat heijastavat roolien ominaisuuksia. Mitä enemmän luetelluista adjektiiveista ja verbeistä olisi roolin ominaisuuksia, sen paremmin kuva esittäisi roolia.

Ensiksi valitsin koehenkilöiden roolikuville antamista adjektiiveista ja verbeistä ominaisuudet, jotka kuvasivat kyseistä roolia. Puolueeton henkilö kirjoitti taulukkoon molemmille samaa roolia esittäville roolikuville annetut ominaisuudet ja lajitteli ne aakkosjärjestykseen. Tämän jälkeen jaoin ominaisuudet suunnilleen kahteen samansuuruiseen ryhmään, joista toinen sisälsi roolin ominaisuuksia kuvaavia adjektiiveja ja verbejä ja toinen ulkopuolelle jääneitä. Tunsin etukäteen rooliteorian sekä olin nähnyt erilaiset roolikuvat ja niille annetut ominaisuudet. Valitettavasti puolueetonta asiantuntijaa ei ollut käytettävissä. Seuraavaksi laskin kuinka suuri joukko kummallekin roolia esittäneelle kuvalle annetuista ominaisuuksista oli roolin ominaisuuksia.

Kuva 13 näyttää kunkin kuvan roolia heijastavien ominaisuuksien määrän prosentteina kaikista kuvalle annetuista ominaisuuksista. Vaaleanharmaat pylväät ovat rooliryhmälle esitettyjen alkuperäisten kuvien prosentteja ja tummemmat pylväät kontrolliryhmän lumekuvien prosentteja. Rooliryhmän kuvista askeltajaa (57 %), kiintoarvoa (62 %) ja kokoojaa (59 %) esittävät kuvat heijastivat roolien ominaisuuksia parhaiten. Ero vastaavaan kontrolliryhmän kuvaan oli myös hyvin selvä. Sen sijaan seuraajaa ja tilapäissäilöä sekä rooli- että kontrolliryhmän kuvat esittivät suunnilleen yhtä hyvin.



Kuva 13: Roolia kuvaavien ominaisuuksien osuus kaikista kuvan ominaisuuksista.

Tilastollisen analyysin tein χ^2 -menetelmällä, jossa vertasin alkuperäisen roolikuvan roolia heijastavien ominaisuuksien määrää lumekuvan roolia heijastavien ominaisuuksien määrään. Askeltajan kohdalla ero kuvien välillä on erittäin merkitsevä ($\chi^2 = 21.305, df = 1, p < 0.0001$). Kiintoarvon analyysissä ero on myös erittäin merkitsevä ($\chi^2 = 12.069, df = 1, p = 0.0005$). Seuraajan osalta tilastollista eroa ei roolikuvan ja lumekuvan välillä ole ($\chi^2 = 0.058, df = 1, p = 0.8099$). Kokoojan kohdalla ero kuvien välillä oli merkitsevä ($\chi^2 = 8.853, df = 1, p = 0.0029$). Tilapäissäilön kohdalla tilastollista eroa ei alkuperäisen roolikuvan ja lumekuvan välillä ole ($\chi^2 = 0.008, df = 1, p = 0.9301$).

Tarkastelen seuraavaksi alkuperäisten PlanAnin roolikuvien ominaisuuksia, koska ne on suunniteltu metaforiksi muuttujien eri rooleille ja siksi niiden pitäisi heijastaa roolien ominaisuuksia. Lumekuvat keksittiin koetta varten vain vertailun vuoksi.

Kuva 14: Askeltaja-roolia esittävät kuvat: a)alkuperäinen kuva b)lumekuva.

Askeltajaa esittänyt roolikuva (kuva 14a) heijasti koehenkilöiden mielestä askeltaja-roolin ominaisuuksia selvästi paremmin kuin kontrolliryhmälle esitetty kuva (kuva 14b). Kaksikymmentäkahdeksan prosenttia askeltajaa esittäneille jalanjäljille annetuista ominaisuuksista liittyi kävelyyn, minkä tulkitsin kuuluvan roolin ominaisuuksiin. Yllättävää oli, että 26 prosenttia ominaisuuksista liittyi epävarmuuteen, mikä on täysin päinvastaista ajatellen askeltajan ennustettavaa ja systemaattista roolikuvaa. Epäröivä, epätietoinen, mietteliäs ja päättämätön olivat sanoja, joilla jalanjälkiä kuvattiin. Todennäköisesti askelten meno eri suuntiin ja siinä välissä pysähtyminen aiheuttivat kuvan yhdistämisen epävarmuuteen. Yksitoista prosenttia jalanjäljille annetuista ominaisuuksista liittyi jotenkin riitaan ja eroamiseen, mikä johtuu ilmeisesti myös jälkien lähtemisestä eri suuntiin.

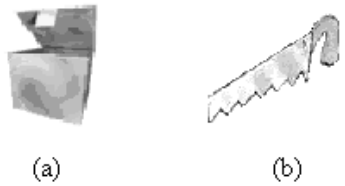
Kuva 15: Kiintoarvo-roolia esittävät kuvat: a)alkuperäinen kuva b)lumekuva.

Kiintoarvoa esittänyt hautakivi (kuva 15a) heijasti roolinsa ominaisuuksia koehenkilöiden mielestä selvästi paremmin kuin vertailukohteena oleva traktorin kuva (kuva 15b). Peräti 38 prosenttia kiintoarvoa esittäneelle hautakivelle annetuista ominaisuuksista liittyi kuolemaan, mitä pidin lopullisuutensa ansiosta kiintoarvo-roolia heijastavana ominaisuutena.

Kuva 16: Seuraaja-roolia esittävät kuvat: a) alkuperäinen b)lumekuva.

Seuraajaa esittävän koiran (kuva 16a) saamat ominaisuudet hajaantuivat niin paljon, että päädyin kategoroimaan ne kolmeen eri sarjaan: koira ystävänä, vartiokoira ja opaskoira. Peräti 47 prosenttia ominaisuuksista kuvasivat koiraa ystävänä, 25 prosenttia vartijana ja 17 prosenttia oppaana. Koira-metafora ei myöskään koehenkilöiden mielestä heijastanut seuraaja-roolin ominaisuuksia sen paremmin kuin vaihtoehtoisena kuvana toiminut kastelukannu (kuva 16b). Vain yksi koehenkilö kuvasi metaforaa sanalla seurata.

Kokoojaa esittävä laatikko (kuva 17a) sai myös osakseen paljon roolia heijastavia ominaisuuksia, selvästi enemmän kuin kontrolliryhmälle kokooja-roolia esittänyt saha (kuva 17b). Kaksikymmentä prosenttia ominaisuuksista liittyi auki olemiseen, mitä pidin



Kuva 17: Kokooja-roolia esittävät kuvat: a)alkuperäinen b)lumekuva.

kokooja-roolin ominaisuutena. Kahdeksantoista prosenttia ominaisuuksista liittyi kiinni olemiseen tai tyhjyyteen, mitä en laskenut roolin ominaisuuksiin. Jonkinlaista epämiellyttävyyttä edusti 10 prosenttia ominaisuuksista. Sanojen joukossa oli esimerkiksi sotkuinen, iljettävä, epäselvä ja häiritsevä. Näitä en pitänyt roolin ominaisuuksina.

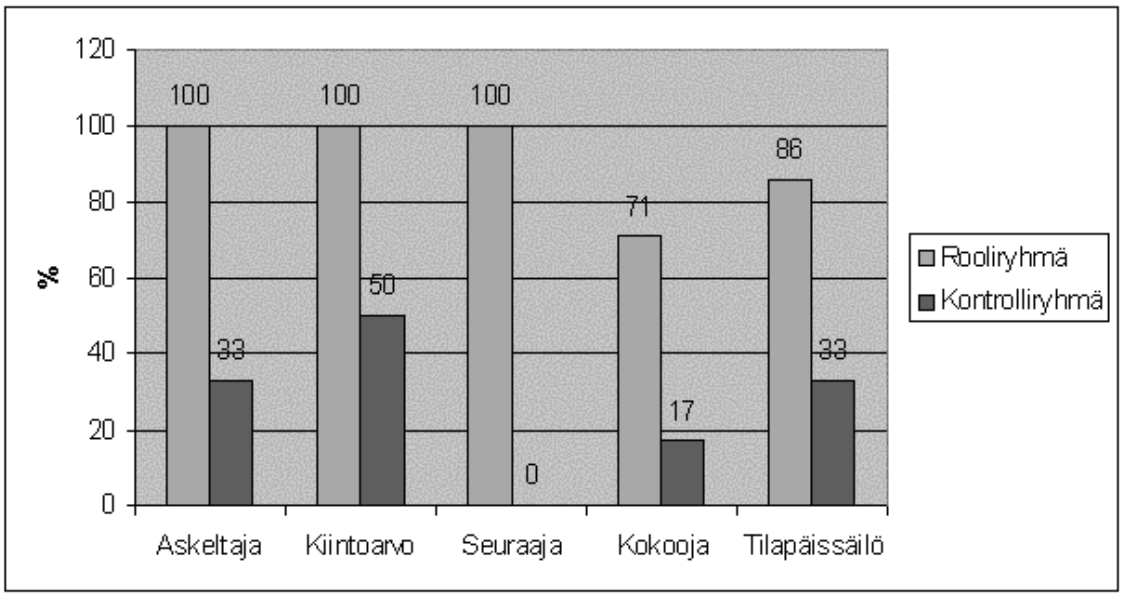
Kuva 18: Tilapäissäilö-roolia esittävät kuvat: a)alkuperäinen b)lumekuva.

Tilapäissäilöä esittävän taskulampun (kuva 18a) saamista ominaisuuksista 42 prosenttia olivat valoon liittyviä sanoja kuten valoisa, kirkas tai hohtava. Yhdeksäntoista prosenttia oli jonkinlaiseen apuvälineeseen liittyviä sanoja kuten näyttää, selventää, opastaa tai neuvoa. Sekä valoon että opastamiseen liittyviä sanoja pidin roolin ominaisuuksina. Taskulamppu-metafora ei koehenkilöiden mielestä heijastanut tilapäissäilö-roolin ominaisuuksia sen paremmin kuin vaihtoehtoinen kävelykeppi-metaforakaan (kuva 18b).

4.2.2 Roolin tunnistaminen kuvasta ja kuvan arviointi

PlanAnilla tapahtuneen roolikuviin tutustumisen jälkeen koehenkilöiden piti tunnistaa kuvasta sen rooli ja arvioida kouluarvosanalla kuinka hyvin kuva esitti roolia. Roolien

tunnistaminen kuvan perusteella laskettiin prosentteina suoraan tehtävän 4 (liitteet 6 ja 7) a-kohdan vastauksista. Arvostelut roolikuville laskettiin keskiarvona tehtävän 4 b-kohdan vastauksista siten, että vain roolin oikein tienneiden arvostelut otettiin mukaan keskiarvoon.



Kuva 19: Eri roolien tunnistaneiden osuus koeryhmästä.

Kuva 19 esittää prosentteina sen kuinka moni koeryhmästä tunnisti kunkin roolin. Vaaleanharmaat pylväät on laskettu rooliryhmän henkilöistä ja tummanharmaat kontrolliryhmäläisistä. Rooliryhmä tunnisti roolit selvästi paremmin kuin kontrolliryhmä. Askeltajan, kiintoarvon ja seuraajan tunnistivat kaikki rooliryhmäläiset kun taas kontrolliryhmästä askeltajan tunnisti vain kolmannes, kiintoarvon tunnisti puolet koehenkilöistä ja seuraajaa ei tunnistanut kukaan. Kokoojan tunnisti rooliryhmästä 71 ja kontrolliryhmästä 17 prosenttia. Tilapäissäilön tunnisti rooliryhmästä 86 ja kontrolliryhmästä 33 prosenttia.

Tilastollisen analyysin tein χ^2 -menetelmän sijasta Fischerin nelikenttätestillä, koska sitä suositellaan käytettäväksi silloin, kun otoskoko on alle 20 tai havaintojen määrä jossakin solussa on alle viisi. Analyysillä vertasin roolikuvan ja lumekuvan välistä eroa roolin tunnistamisessa kuvan perusteella. Askeltajan kohdalla ero oli merkitsevä ($p=0.0210$), kiintoarvon kohdalla ero oli melkein merkitsevä ($p=0.0699$), seuraajan kohdalla ero oli erittäin merkitsevä ($p=0.0006$), kokoojan kohdalla ero oli melkein merkitsevä ($p=0.0775$) ja tilapäissäilön kohdalla myös melkein merkitsevä ($p=0.0862$).

Taulukossa 3 näkyy rooleittain koehenkilöiden antamien kouluarvosanojen keskiarvot. Askeltaja, kiintoarvo ja kokooja arvioitiin rooliryhmän toimesta kiitettäviksi (8.8 - 9.3), seuraaja tyydyttäväksi (8) ja tilapäissäilö välttäväksi (5.8). Kontrolliryhmä arvioi kiintoarvon tyydyttäväksi (6.7) ja muut välttäviksi (0 - 5.5).

Tilastollisen analyysin tein kaksisuuntaisella t-testillä, jolla vertasin rooli- ja kontrolliryhmien rooleja esittäville kuville antamia kouluarvosanoja. Askeltajan kohdalla ero on testin mukaan erittäin merkitsevä ($t = 5.131, df = 7, p = 0.0014$) ja kiintoarvon kohdalla ero on merkitsevä ($t = 2.909, df = 8, p = 0.0196$). Tilapäissäilön roolikuvan ja lumekuvan saamien arvosanojen välillä tilastollista eroa ei ole ($t = 0.548, df = 6, p = 0.6036$). Seuraajan ja kokoojan osalta tilastollista analyysiä ei voitu tehdä, koska kontrolliryhmästä yksikään ei tunnistanut seuraajaa lumekuvasta ja vain yksi tunnisti kokoojan.

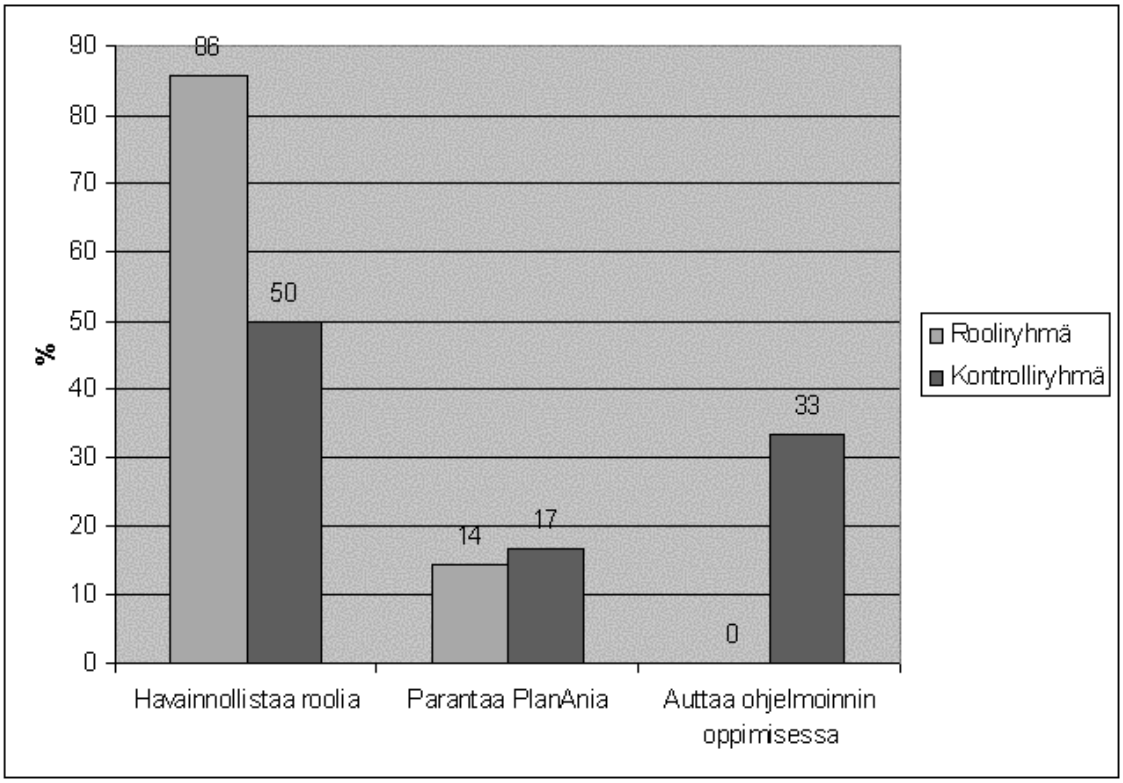
Koehenkilöt saivat myös ehdottaa oman vaihtoehtonsa roolia kympin arvoisesti esittäväksi kuvaksi. Varsinkin kontrolliryhmä oli innokas keksimään vaihtoehtoisia kuvia, mikä saattoi ainakin osittain johtua lumekuvien huonosta roolin esittämisestä. Askeltajaa esittäväksi metaforaksi kontrolliryhmän koehenkilöt ehdottivat kävelevää ihmistä, portaikkoa tai tuhatjalkaista. Rooliryhmän koehenkilöt pitivät kuvaa niin hyvänä, etteivät ehdottaneet tilalle mitään muuta. Vaihtoehdoksi kiintoarvoa esittäneelle kuvalle koehenkilöt ehdottivat kalliota, kiveä, pyramidia, puuta tai aurinkoa. Koehenkilöiden ehdotuksia seuraajaa esittäväksi kuvaksi olivat varjo, äitiä seuraava lapsi, jälkikoira tai jälkiä seuraava ukko. Koehenkilöiden ehdotuksia kokoojaa esittäväksi kuvaksi olivat palapelin palat, kokoontumispaikka, jäteastia, rakennusnosturi, kori sekä laatikko, jonka sisällä olisi jotain. Parempia kuvia tilapäissäilöä esittämään olisivat koehenkilöiden mielestä esimerkiksi muovipussi, lintuhäkki, hyllyltä toiselle siirtyvät tavarat, marketin säilytyslokero, WC-istuin, kastelukannu, lasi, hattu alassuin tai kertakäyttöastia.

Taulukko 3: Roolikuville annettujen kouluarvosanojen (4-10) keskiarvot.

Koeryhmä	*Askeltaja*	*Kiintoarvo*	*Seuraaja*	*Kokooja*	*Tilapäissäilö*
Rooliryhmä	9.3	8.9	8	8.8	5.8
Kontrolliryhmä	5.5	6.7	0	5	5

4.2.3 Roolikuvien käytön tarpeellisuus

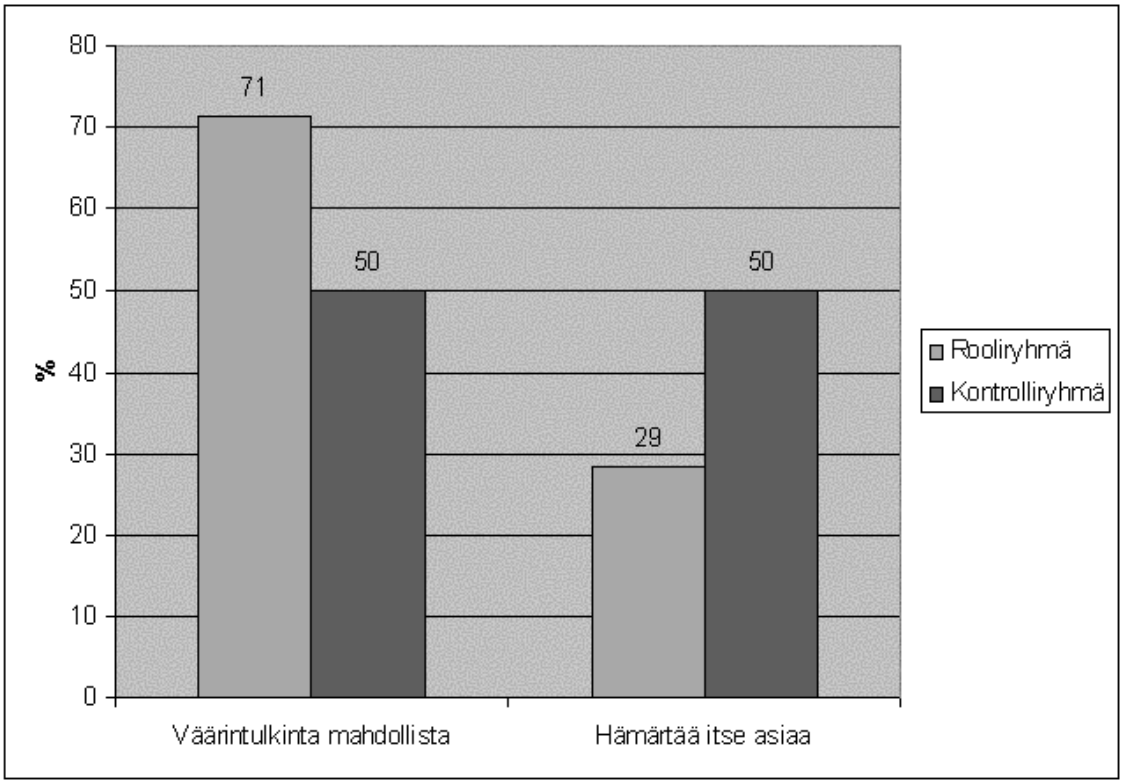
Koehenkilöitä pyydettiin arvioimaan kouluarvosanalla kuinka hyödyllistä roolien esittäminen kuvilla on. Roolikuvien hyödyllisyyden laskin koehenkilöiden antamista kouluarvosanoista tehtävän 5 (liite 8) a-kohdan perusteella. Kuvien hyödyt kategorisoin kolmeen ryhmään: havainnollistaa roolia, parantaa PlanAnia ja auttaa ohjelmoinnin oppimisessa. Kuvien haitat kategorisoin kahteen ryhmään: väärintulkinta mahdollista ja hämärtää itse asiaa. Sen jälkeen laskin koehenkilöiden osuudet kustakin kategoriasta prosentteina koeryhmän sisällä.



Kuva 20: Roolien esittäminen kuvilla: hyödyt.

Rooliryhmän koehenkilöt antoivat hyödyllisyydelle keskiarvon 8.2 ja kontrolliryhmäläiset arvostivat kuvan käyttöä keskiarvolla 7.5. Kaksisuuntainen t-testi osoittaa, että ero rymien välillä ei ole merkitsevä ($t = 1.207, df = 11, p = 0.2527$). Kuva 20 esittää kuinka suuri prosentuaalinen osuus koeryhmästä arvioi kuvilla esittämisen hyötyjä milläkin tavalla. Rooliryhmän mielipiteet esitetään vaaleanharmaina ja kontrolliryhmän tummanharmaina pylväinä. Rooliryhmän mielestä (86 %) suurin hyöty kuvan käytöstä oli sen kyky havainnollistaa roolia. Myös kontrolliryhmän mielestä se oli suurin kuvan käytön hyöty, mutta heistä vain puolet oli tätä mieltä. Molemmat

koeryhmät olivat tasaprosentein sitä mieltä, että kuvien käyttö parantaa PlanAnia. Kontrolliryhmästä 33 prosenttia piti hyötynä sitä, että kuvat auttavat ohjelmoinnin oppimisessa.



Kuva 21: Roolien esittäminen kuvilla: haitat.

Kuvassa 21 esitetään kuvilla esittämisen haittoja. Rooliryhmän mielestä suurin haitta (71 %) on kuvien väärintulkinnan mahdollisuus. Kontrolliryhmästäkin 50 prosenttia oli tätä mieltä. Rooliryhmästä 29 prosenttia oli sitä mieltä, että kuvien käyttö voi hämärtää itse asiaa eli tässä tapauksessa ohjelmoinnin oppimista. Kontrolliryhmästä jopa puolet oli tätä mieltä.

4.2.4 Roolien tunnistaminen ohjelmista

Koehenkilöt saivat roolien kertauksen jälkeen tehdä tasokokeen, jonka perusteella koehenkilöt jaettiin kahteen tasavahvaan ryhmään. Tehtävänä oli tunnistaa muuttujien rooleja kolmesta pienestä Pascal-kielisestä ohjelmasta. Viimeisenä kokeen tehtävänä oli

Taulukko 4: Vertailu roolien tunnistuksesta kokeen alussa ja lopussa.

Koeryhmä	*Alussa (max 14)*	*Vastattuja*	*Lopussa (max 14)*	*Vastattuja*	*Ero*
Rooliryhmä	6.7	9.7	11.4	14	4.7
Kontrolliryhmä	6.5	9.8	5.8	10.8	-0.7

myös roolien tunnistus kolmesta ohjelmasta. Roolien tunnistamisessa ohjelmista tapahtuneen eron laskin kokeen alussa olleen tasokokeen (liite 2; Sajaniemi, 2004) ja kokeen viimeisen tehtävän (liite 9) perusteella. Molemmissa oli maksimipistemäärä 14 sen jälkeen, kun viimeisestä tehtävästä jätettiin arvostelematta kaksi ei tyypillisessä tilanteessa käytettyä roolia: a-kohdan firstPlayer on askeltaja ja b-kohdan amount on kiintoarvo. Sekä tasokokeesta että kokeen viimeisestä tehtävästä laskin koeryhmien keskiarvot sekä vastattujen tehtävien määrien keskiarvot.

Taulukossa 4 esitetään koeryhmien roolien tunnistustehtävistä saamat pisteet sekä vastattujen tehtävien määrät. Rooliryhmän koehenkilöt tunnistivat viimeisessä tehtävässä selvästi enemmän rooleja (4.7) kuin tasokokeessa ja vastattujen tehtävien määrä osoittaa myös motivaation olleen parempi kuin kontrolliryhmällä. Kontrolliryhmä tunnisti rooleja viimeisessä tehtävässä hieman vähemmän (-0.7) kuin tasokokeessa ja vastattujen tehtävien määrä nousi vain hieman alkutilanteesta.

Tilastollisen analyysin tein Wilcoxonin testillä, joka on parittaisen t-testin eiparametrinen vastine. Wilcoxonin testiä voidaan käyttää, vaikka parittaiset erotukset eivät noudattaisikaan normaalijakaumaa, mutta rajoituksena on, että populaation jakauman täytyy olla symmetrinen. Tässä tapauksessa sekä rooliryhmän että kontrolliryhmän jakaumat ovat symmetrisiä, mutta normaalijakaumasta ei voida puhua, koska toisen tulokset ovat pieniä ja toisen suuria. Analyysillä vertasin siis rooli- ja kontrolliryhmän eroa roolien tunnistamisessa ohjelmakoodista. Testin mukaan ero on erittäin merkitsevä ($p = 0.0012$).

4.3 Tarkastelu

Tarkastelen jokaista roolikuvaa erikseen rinnastaen sen eri koeosioista saamia tuloksia ja verraten niitä vastaavaan lumekuvaan ja sen tuloksiin. Jokaisesta kuvasta saatiin tietoa koehenkilöiltä jo ennen kuin he näkivät sen roolin yhteydessä sekä sen jälkeen,

kun sitä käytettiin PlanAnissa roolin metaforana ja lisäksi koehenkilöt antoivat kuvalle arvosanan roolin esittämisestä. Kokeella on myös mitattu miten hyvin koehenkilöt tunnistavat rooleja ohjelmista kokeen alussa ja lopussa ja koehenkilöt ovat saaneet antaa palautetta siitä miten hyödyllistä roolin esittäminen kuvalla on.

4.3.1 Askeltaja

Askeltajaa esittäneelle roolikuvalle (kuva 14a) koehenkilöt luettelivat askeltajan rooliominaisuuksia 57 prosenttia ja muita loput 43 prosenttia. Askeltajaa esittänyt lumekuva (kuva 14b) sai rooliominaisuuksia vain 8 prosenttia ja loput 92 prosenttia muita kuin roolin ominaisuuksia. Tilastollisestikin kuvien ero on erittäin merkitsevä ($p < 0.0001$). Puutteeksi askeltajaa esittävälle metaforalle mielestäni jää, ettei se koehenkilöiden vastausten perusteella heijasta ennustettavuutta, systemaattisuutta eikä järjestelmällisyyttä, mitkä ovat olennaisia askeltaja-roolin ominaisuuksia.

Kun askeltaja-rooli piti tunnistaa kuvan perusteella, tunnistivat kaikki rooliryhmän koehenkilöt askeltajan, kun kontrolliryhmässä askeltajan tunnisti lumekuvasta vain kolmannes koehenkilöistä. Ero on tilastollisesti merkitsevä ($p = 0.0210$). Roolikuvan tunnistamisessa helpotti todennäköisesti myös askeltajan kirjallinen metafora eli nimi. Muistitutkimukset ovat osoittaneet, että nimeen liittyvät mielikuvat helpottavat muistamista (Paivio & Walsh, 1993); esimerkiksi nimi on helpompi muistaa, kun siihen liittyy kuva. Tässä tapauksessa askeltaja-roolista tuli mieleen jalanjäljet, joten oikea rooli löytyi helposti. Askeltaja-rooli ei millään tavalla viitannut nimellään lumekuvana toimineeseen kirveeseen, joten sen roolia ei pystynyt arvaamaan.

Rooliryhmän mielestä jalanjälki-metafora kuvaa askeltajaa kiitettävästi arvosanalla 9.3. Kirves-metafora kuvaa kontrolliryhmän mielestä askeltajaa vain välttävästi arvosanalla 5.5. Tilastollisesti ero on erittäin merkitsevä ($p = 0.0014$).

Edellisen perusteella askeltajan alkuperäinen roolikuva heijastaa paremmin roolin ominaisuuksia, on paremmin tunnistettavissa askeltajaksi ja koehenkilöiden mielestä kuvaa paremmin rooliaan kuin vaihtoehtoinen kirves-kuva. Tosin askeltaja-nimen vaikutuksen suuruutta tuloksiin on vaikeaa arvioida. Lisäksi jalanjälki-kuvaa täytyisi hieman kehittää, jotta siitä ei saisi epäröivää mielikuvaa vaan se kuvastaisi enemmänkin päättäväistä ja ennustettavaa kulkua.

4.3.2 Kiintoarvo

Kiintoarvoa esittäneelle roolikuvalle (kuva 15a) koehenkilöt listasivat 62 prosenttia roolin ominaisuuksia, kun taas lumekuva (kuva 15b) sai vain 27 prosenttia roolin ominaisuuksia ja valtaosan muita. Tilastollisesti ero kuvien välillä on erittäin merkitsevä ($p = 0.0005$). Mitään erityisiä puutteita tai ristiriitaisuuksia hautakivi-metaforalle ei kokeen perusteella ilmennyt.

Rooliryhmästä kaikki henkilöt tunnistivat hautakivestä roolin kiintoarvo. Kontrolliryhmästä kiintoarvon tunnistivat traktori-kuvasta puolet koehenkilöistä. Tilastollisesti ero on melkein merkitsevä ($p = 0.0699$).

Hautakivi-metafora sai rooliryhmältä kiitettävän arvosanan 8.9, kun lumekuvana toiminut traktori sai kontrolliryhmältä arvosanan 6.7. Tilastollisesti eroa arvosanojen välillä on merkitsevästi ($p = 0.0196$).

Hautakivi heijasti enemmän roolinsa ominaisuuksia kuin lumekuva ja jäi koehenkilöille paremmin mieleen. Koehenkilöt myös pitivät hautakiveä kuvaavampana metaforana kuin traktoria. Tämän kokeen perusteella näyttäisi siltä, että kiintoarvo-roolia kuvaa paremmin alkuperäinen roolikuva.

4.3.3 Seuraaja

Seuraajaa esittäneelle alkuperäiselle roolikuvalle (kuva 16a) koehenkilöt antoivat 42 prosenttia roolin ominaisuuksia ja suurimman osan (58 %) muita. Lumekuvana toiminut kastelukannu (kuva 16b) sai lähes saman verran eli 39 prosenttia roolin ominaisuuksia ja 61 prosenttia muita. Tilastollisesti eroa kuvien välillä ei ole ($p = 0.8099$).

Seuraajaa esittänyt koira tunnistettiin täydellisesti seuraajaksi ja lumekuvana toiminutta kastelukannua ei tunnistettu ollenkaan seuraajaksi, joten tilastollinen erokin on erittäin merkitsevä ($p = 0.0006$). Luulen, että tunnistamisessa kuitenkin auttoi jälleen roolin nimi seuraaja. Koiraa voidaan komentaa ”seuraa”-komennolla, joten sanaan liittää helposti koiran (Paivio & Walsh, 1993). Valittaessa koiran kuvalle roolia askeltajan, järjestelijän, kiintoarvon, kokoojan, seuraajan, sopivimman säilyttäjän, tilapäissäilön, tuoreimman säilyttäjän tai yksisuuntaisen lipun joukosta, osuu valinta helposti seuraajaan. Lumekuvana toiminut kastelukannu sekoitettiin sekä kokoojaan että tilapäissäilöön todennäköisesti siksi, että kumpikin nimi voisi viitata kastelukannu-kuvaan.

Koira-metafora sai rooliryhmältä arvosanaksi tyydyttävän 8 ja kastelukannu kontrolliryhmältä pyöreän nollan, mikä johtuu siitä, että laskin keskiarvoon vain roolin oikein tunnistaneiden antamat arvosanat. Tilastollista analyysiä ei voitu tehdä, koska kukaan kontrolliryhmästä ei tunnistanut kastelukannua seuraajaksi.

On ristiriitaista, että seuraajan koira-metafora tunnistettiin paremmin, mutta sai osakseen suunnilleen saman verran roolin ominaisuuksia kuin kastelukannu-metaforakin. Mielestäni parempi tunnistaminen ei tässä tapauksessa johdu koira-metaforan sopivuudesta esittämään seuraaja-roolia, vaan roolinimen ohjaavasta vaikutuksesta. Seuraajaa esittänyt koira ei mielestäni tämän kokeen perusteella kuvaa roolia sen paremmin kuin lumekuvana toiminut kastelukannukaan. Koehenkilöiden ehdotuksia seuraajaa kuvaaviksi metaforiksi olivat esimerkiksi varjo, äitiä seuraava lapsi, jälkikoira tai jälkiä seuraava ukko.

4.3.4 Kokooja

Kokoojaa esittäneelle alkuperäiselle roolikuvalle (kuva 17a) rooliryhmä antoi 59 prosenttia roolin ominaisuuksia kun taas lumekuva (kuva 17b) sai vain 29 prosenttia roolin ominaisuuksia ja 71 prosenttia muita. Tilastollisestikin ero on merkitsevä ($p = 0.0029$). Jonkinlaista epämiellyttävyyttä edusti 10 prosenttia kokoojana toimineen laatikon ominaisuuksista, mikä oli mielestäni yllättävää. Kuva saattoi olla koehenkilöiden mielestä jollakin tapaa epäselvä. Yksikään koehenkilö ei ehdottanut laatikko-kuvan ominaisuudeksi mitään summaamiseen, kumuloimiseen tai yhdistämiseen viittaavaa sanaa, mitä pidän metaforan puutteena. Kokooja-roolihan nimenomaan saa arvonsa summaamalla tai kumuloimalla muiden muuttujien arvoja.

Kokooja-roolin tunnisti pahvilaatikko-kuvasta 71 prosenttia rooliryhmän henkilöistä, kun lumekuvana toimineesta sahasta kokoojan tunnisti vain 17 prosenttia kontrolliryhmäläisistä. Ero on tilastollisesti melkein merkitsevä ($p = 0.0775$). Mikäli pahvilaatikko-kuva oli koehenkilöistä epäselvä, on ymmärrettävää ettei sitä tunnistettu kokoojaksi paremmin. Nimen perusteella kokooja-rooli sekoitettiin tilapäissäiliöön, koska kuva olisi hyvin voinut esittää sitäkin.

Rooliryhmä antoi kokoojaa esittäneelle pahvilaatikko-metaforalle kouluarvosanaksi kiitettävän 8.8. Kontrolliryhmästä vain yksi koehenkilö tunnisti kokooja-roolin sahakuvasta ja antoi tälle välttävän arvosanan 5. Koska kontrolliryhmästä saatiin arvosanoja niin vähän, ei tilastollista analyysiä voitu tehdä.

Pahvilaatikko-kuvaa voidaan tämän kokeen perusteella pitää paremmin kokoojaa kuvaavana metaforana kuin saha-kuvaa. Kuvaa pitää kuitenkin hieman kehittää, ettei se herättäisi epämiellyttävyyden ja epäselvyyden tuntemuksia. Olisi hyvä myös saada se heijastamaan roolinsa olennaista ominaisuutta summaamista tai kumuloimista. Kokooja tunnistettiin hyvin, mutta myös sekoitettiin tilapäissäilöön, mikä viittaa nimen ohjaavaan vaikutukseen. Kuvan kehittämisessä voisi myös huomioida tilapäissäilöön sekoittumisen vaaran.

4.3.5 Tilapäissäilö

Tilapäissäilöä esittänyt alkuperäinen roolikuva, taskulamppu (kuva 18a), sai rooliryhmältä 48 prosenttia roolin ominaisuuksia ja loput 52 prosenttia muita. Lumekuvana toiminut kävelykeppi (kuva 18b) sai jopa hiukan enemmän (49 %) roolin ominaisuuksia. Tilastollisesti kuvien välillä ei ollut eroa, joten alkuperäinen roolikuva ei heijasta tilapäissäilö-roolin ominaisuuksia sen paremmin kuin lumekuvana toiminut kävelykeppikään. Tilapäissäilöä esittävän taskulamppu metaforan puutteena pidän sitä, ettei se heijastanut roolin pääasiallista tarkoitusta eli tilapäisyyttä.

Taskulamppu tunnistettiin kuitenkin tilapäissäilöksi paremmin (86 %) kuin kävelykeppi (33 %). Tilastollisesti ero on melkein merkitsevä ($p = 0.0862$). Parempi tunnistettavuus saattoi johtua PlanAnissa käytetystä animaatiosta, jossa korostettiin nimenomaan roolin tilapäisyyttä sytyttämällä ja sammuttamalla taskulamppu. Metaforan ominaisuuksia lueteltiin staattisen kuvan perusteella, jossa lamppu oli koko ajan päällä, joten siinä tilapäisyyttä ei tullut esille.

Rooliryhmä antoi taskulamppu-metaforalle arvosanaksi välttävän 5.8 ja kontrolliryhmä kävelykeppi-metaforalle myös välttävän 5, joten koehenkilöiden mielestä kumpikaan ei kuvannut tilapäissäilöä kunnolla. Tilastollista eroa kuvien välillä ei myöskään ole ($p = 0.6036$).

Tämän kokeen perusteella taskulamppua ei voida pitää kävelykeppiä parempana metaforana kuvaamaan tilapäissäilö-roolia. Vaihtoehdoiksi metaforalle koehenkilöt esittivät esimerkiksi muovipussia, lintuhäkkiä, marketin säilytyslokeroa, WC-istuinta, kastelukannua ja lasia.

4.3.6 Rooliteorian omaksuminen

Tämän kokeen perusteella vaikuttaa siltä, että alkuperäiset askeltajaa, kiintoarvoa ja kokoojaa esittävät roolikuvat välittävät roolien ominaisuuksia koehenkilöille paremmin kuin vaihtoehtoiset roolikuvat. Seuraajaa ja tilapäissäilöä esittävät alkuperäiset roolikuvat taas eivät heijasta roolien ominaisuuksia lumekuvia paremmin.

Rooliryhmän taitojen kehittyminen kokeen kuluessa voisi johtua siitä, että rooleja esittäneet metaforat ovat edistäneet rooliteorian omaksumista. Kontrolliryhmällä positiivista kehitystä ei tapahtunut, joten mielestäni kehitystä ei voi selittää pelkällä rooliteorian aktivoitumisella kokeen kuluessa. Molemmat koeryhmät tekivät samat tehtävät, vain rooleja esittäneet kuvat olivat erilaisia.

Molemmat koeryhmät antoivat tyydyttävän arvosanan roolien kuvallisen esittämisen hyödyllisyydelle. Molempien koeryhmien mielestä parhaita kuvallisen ilmaisun etuja oli roolin havainnollistaminen ja suurimpana haittana he pitivät mahdollista kuvan väärintulkintaa. Tästäkin voidaan päätellä, että kuvalla on väliä ja että metaforan valintaan täytyy kiinnittää paljon huomiota, jotta sen viesti menisi perille mahdollisimman oikeanlaisena.

5 Yhteenveto

Metaforan avulla pystytään selittämään abstrakti käsite tai monimutkainen asia toisen konkreettisemman ja yksinkertaisemman asian avulla (Lakoff, 1993). PlanAni-ohjelma-animaattorin tehtävänä on havainnollistaa aloitteleville ohjelmoijille yksinkertaisen ohjelman suoritusta. Havainnollistaminen perustuu Sajaniemen (2002a) muuttujien rooliteoriaan, mikä tarkoittaa, että jokaista roolia esitetään kuvalla ja muuttujan arvon vaihtuminen ja vertailu havainnollistetaan roolikohtaisilla animaatioilla. Rooleja esittävät kuvat toimivat metaforina rooleille ja ne on pyritty valitsemaan siten, että ne välittäisivät opiskelijoille roolien tärkeimpiä ominaisuuksia.

Halusin tutkia miten hyvin roolikuvat toimivat metaforina ja helpottavat rooliteorian omaksumista. Tutkimusmenetelmänä käytin Alty *& al:*in metaforien valintaan, suunnitteluun ja toteutukseen kehittämää ohjeistoa. Koetta varten piti keksiä myös vaihtoehtoiset roolikuvat, jotta alkuperäisten kuvien saamia tuloksia voitaisiin vertailla johonkin. Lumekuvat valitsin siten, että ne kuvasivat mahdollisimman huonosti roolinsa ominaisuuksia. Näin kuvien välille saatiin selviä eroja niissä tapauksissa, kun alkuperäiset roolikuvat olivat suhteellisen hyvin toimivia. Huonommin toimivat kuvat saivat yhtä huonoja tuloksia kuin niitä vastaavat lumekuvat. Koetehtävien tulosten perusteella pystyin myös päättelemään kuinka hyvin toimivia kuvia voisi edelleen kehittää, jotta niistä saataisiin vieläkin parempia.

Tutkimuksen kohteena oli viisi roolia: askeltaja, kiintoarvo, seuraaja, kokooja ja tilapäissäilö. Tulosten perusteella askeltajaa, kiintoarvoa ja kokoojaa esittävät kuvat toimivat hyvin kuvallisina metaforina rooleilleen. Askeltajan roolikuvaa voisi kehittää siten, ettei siitä saisi epäröivää mielikuvaa, vaan se kuvastaisi enemmänkin ennustettavaa kulkua. Jotkut koehenkilöistä pitivät kokoojaa esittävää pahvilaatikkoa epämiellyttävänä eikä yksikään yhdistänyt siihen sen olennaisimpia ominaisuuksia eli summaamista tai kumuloimista. Pahvilaatikko myös sekoitettiin tilapäissäilöön, mikä pitäisi ottaa huomioon kuvia edelleen kehitettäessä. Seuraajaa ja tilapäissäilöä esittävät kuvat eivät toimineet lumekuvia paremmin. Koehenkilöiden ehdotuksia seuraajaa paremmin esittäväksi kuvaksi olivat esimerkiksi varjo, äitiä seuraava lapsi, jälkikoira tai jälkiä seuraava ukko. Tilapäissäilöä esittäisi koehenkilöiden mielestä paremmin esimerkiksi muovipussi, lintuhäkki, marketin säilytyslokero, WC-istuin, kastelukannu tai lasi.

Mielestäni kokeella saatiin hyvin esille roolikuvien ominaisuuksia ja useiden tehtävien tulokset vahvistivat toisiaan, mikä lisäsi luottamusta kokeen tuloksiin. Lisäksi koehenkilöiden kuville antamat arvosanat ja ehdotukset rooleja paremmin esittäviksi kuviksi ovat arvokasta materiaalia kuvia edelleen kehitettäessä. Tässä tutkittiin vain puolet roolikuvista eikä roolianimaatioiden vaikutusta kuvien toimivuuteen otettu ollenkaan huomioon, joten jatkotutkimusta vielä tarvitaan.

Viitteet

Alty, J. L., Knott, R. P., Anderson, B., Smyth, M. (2000) A Framework for Engineering Metaphor at the User Interface. *Interacting with Computers* **13**, 301-322.

Ben-Ari, M., Sajaniemi, J. (2004) Roles of Variables as Seen by CS Educators. *Accepted to the 9th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2004).* Leeds, UK, 28-30.

Carrol, J.M., Mack, R.L. (1999) Metaphor, Computing Systems, and Active Learning. *International Journal of Human-Computer Studies* **51**, 385-403.

Glucksberg, S., Keysar, B. (1993) How Metaphors Work. *Metaphor and Thought* (toim. Ortony, A.), Press Syndicate of the University of Cambridge, Cambridge, 401-424.

Hamilton, A. (1989) Interface Metaphors and Logical Analogues: A Question of Terminology. *Journal of the American Society for Information Science* **51**(2), 111-122.

Helsingin sanomat (2004) *Viivi ja Wagner*. WWW-sivusto, `http://www.helsinginsanomat.fi/viivijawagner/1052754110580` (11.5.2004).

Kielikone Oy (2003) *MOT Sanakirjasto*. WWW-sivusto, `http://80-mot.kielikone.fi.joecat.joensuu.fi:8080/mot/joyo/netmot.exe/` (18.10.2003).

Lakoff, G. (1993) The Contemporary Theory of Metaphor. *Metaphor and Thought* (toim. Ortony, A.), Press Syndicate of the University of Cambridge, Cambridge, 202-251.

Lakoff, G., Johnson, M. (1980) *Metaphors We Live by*. University of Chicago Press, Chicago.

Paivio, A., Walsh, M. (1993) Psychological Processes in Metaphor Comprehension and Memory. *Metaphor and Thought* (toim. Ortony, A.), Press Syndicate of the University of Cambridge, Cambridge, 307-328.

Sajaniemi, J. (2002a) An Empirical Analysis of Roles of Variables in Novice-Level Procedural Programs. *Proceedings of IEEE 2002 Symposia on Human Centric Computing Languages and Environments (HCC'02).* IEEE Computer Society, 37-39.

Sajaniemi, J. (2002b) Visualizing Roles of Variables to Novice Programmers. *Proceedings of the Fourteenth Annual Workshop of the Psychology of Programming Interest Group (PPIG 2002)* (toim. Kuljis, J., Baldwin, L., Scoble, R.), London, UK, 111-127.

Sajaniemi, J. (2004) *Roles of Variables*. WWW-sivusto, `http://www.cs.joensuu.fi/~saja/var_roles/teaching.html` (3.2.2004).

Sajaniemi, J., Kuittinen, M. (2003) Program Animation Based on the Roles of Variables. *Proceedings ACM 2003 Symposium on Software Visualization (SoftVis 2003).* Association for Computing Machinery, 7-16.

Sajaniemi, J., Kuittinen, M. (in press) An Experiment on Using Roles of Variables in Teaching Introductory Programming. *Computer Science Education.*

Wozny, L.A. (1989) The Application of Metaphor, Analogy and Conceptual Models in Computer Systems. *Interacting with Computers* **1**(3), 273-283.

Muuttujien roolit

Liite 1

Muuttujilla on ohjelmissa eräitä tyypillisiä käyttötapoja, joita kutsutaan muuttujien *rooleiksi*. Esimerkiksi muuttuja, jonka arvoa ei muuteta enää kertaakaan muuttujan alustamisen jälkeen, kutsutaan *kiintoarvoksi*. Muuttujien rooleja ei pidä sekoittaa muuttujien perustietotyyppeihin (Pascalissa esimerkiksi integer, char, boolean jne.), vaan kyseessä on jaottelu, joka ilmaisee millainen tehtävä muuttujalla kyseisessä ohjelmassa on.

Seuraavassa kuvattavat yhdeksän eri roolia riittävät kattamaan lähes kaikki yksinkertaisissa ohjelmissa esiintyvät muuttujat. Useimmiten esiintyvät roolit ovat *kiintoarvo*, *askeltaja* ja *tuoreimman säilyttäjä*, jotka kattavat noin 80 % kaikista muuttujista. Toisaalta on huomattava, että kaikilla muuttujilla ei välttämättä ole mitään alla esitellyistä rooleista.

Muuttujien roolit, jotka on esitelty tarkemmin alla, ovat

- *kiintoarvo*
- *askeltaja*
- *tuoreimman säilyttäjä*
- *sopivimman säilyttäjä*
- *kokooja*
- *yksisuuntainen lippu*
- *seuraaja*
- *tilapäissäilö*
- *järjestelijä*

Esimerkkiohjelmien yhteydessä olevan rivinumeroinnin tarkoitus on helpottaa riveihin viittaamista. Ohjelmien suorituksen kannalta rivinumeroinnilla ei ole mitään merkitystä, koska ne ovat muodoltaan Pascal-kommentteja.

Kiintoarvo

Muuttujan rooli on *kiintoarvo*, jos sen arvoa ei muuteta ohjelman suorituksen aikana muuttujan alustamisen jälkeen. Esimerkkiohjelma kysyy käyttäjältä ympyrän säteen ja ilmoittaa sitten ympyrän alan. Muuttuja `r` on *kiintoarvo*. Kyseinen muuttuja saa ohjelman suorituksen aikana yhden kerran arvon (rivillä 7), joka ei sen jälkeen muutu. Kiintoarvoa voidaan käyttää ohjelman eri kohdissa - esimerkissä kahdesti rivillä 8.

```
(*1*)  program YmpyranAla (input,output);
(*2*)  const PII = 3.14;
(*3*)  var r: real;
(*4*)  begin
(*5*)     writeln;
(*6*)     write ('Anna ympyrän säde: ');
(*7*)     readln (r);
(*8*)     writeln ('Ympyrän ala on ', PII * r * r)
(*9*)  end.
```

Askeltaja

Askeltaja käy läpi arvoja jollain systemaattisella tavalla. Alla on esimerkki silmukkarakenteesta, jossa käytetään muuttujaa `kertoja` *askeltajana*. Esimerkkiohjelma tulostaa kolmosen kertotaulun *askeltajan* käydessä läpi arvot yhdestä kymmeneen.

```
(*1*) program Kertotaulu (output);
(*2*) var kertoja: integer;
(*3*) begin
(*4*)    for kertoja := 1 to 10 do
(*5*)        writeln(kertoja, ' * 3 = ', kertoja*3)
(*6*) end.
```

Askeltajaa voidaan käyttää myös esimerkiksi lukumäärän laskemiseen ja taulukon indeksien läpikäymiseen.

Tuoreimman säilyttäjä

Muuttuja on *tuoreimman säilyttäjä*, jos sen arvo on viimeisin jostakin tietystä joukosta läpikäyty arvo tai yksinkertaisesti vain arvo, joka on syötetty viimeksi. Esimerkkiohjelma pyytää käyttäjältä syötettä toistuvasti (rivillä 6) kunnes syöte on kelvollinen. Tässä ohjelmassa muuttuja `s` on *tuoreimman säilyttäjä*, koska siitä löytyy kulloinkin viimeksi syötetty arvo.

```
(*1*)  program NelionAla (input,output);
(*2*)  var s: real;
(*3*)  begin
(*4*)     repeat
(*5*)        write ('Anna neliön sivu: ');
(*6*)        readln (s)
(*7*)     until s > 0;
(*8*)     writeln('Neliön ala on ', s * s)
(*9*)  end.
```

Sopivimman säilyttäjä

Sopivimman säilyttäjän arvo on "paras" tai jollain muulla tavoin halutuin siihen asti läpikäydyistä arvoista. Arvojen paremmuuden mittaamisessa ei ole mitään rajoituksia: halutuin voi tarkoittaa esimerkiksi pienintä tai suurinta lukua tai sellaista lukua, joka on lähinnä jotain tiettyä arvoa.

Esimerkkiohjelma selvittää, mikä käyttäjän syöttämistä kymmenestä kokonaisluvusta on pienin. Muuttuja `pienin` on *sopivimman säilyttäjä*, koska siihen sijoitetaan (rivillä 8) tuorein arvo, mikäli se on pienempi kuin pienin tähän mennessä läpikäydyistä.

```
(*1*)  program EtsiPienin (input,output);
(*2*)  var i, pienin, luku: integer;
(*3*)  begin
(*4*)     write('Anna 1. luku: '); readln(pienin);
(*5*)     for i := 2 to 10 do begin
(*6*)        write('Anna ',i,'. luku: ');
(*7*)        readln(luku);
(*8*)        if luku < pienin then pienin := luku
(*9*)     end;
(*10*)    writeln ('Pienin luku oli ', pienin)
(*11*) end.
```

(Muuttuja `i` on *askeltaja* ja `luku` on *tuoreimman säilyttäjä*.)

Kokooja

Kokoojan arvo kerääntyy kaikista siihen mennessä läpikäydyistä arvoista. Esimerkkiohjelma ottaa vastaan yksi kerrallaan käyttäjän syöttämiä kokonaislukuja, kunnes käyttäjä syöttää luvun -999, jonka jälkeen ohjelma laskee syötteiden keskiarvon. Muuttuja `summa` on *kokooja*: siihen kootaan (rivillä 10) syötteiden kokonaissummaa.

```
(*1*)  program Keskiarvo (input,output);
(*2*)  var lkm: integer;
(*3*)      summa, luku: real;
(*4*)  begin
(*5*)     summa := 0;
(*6*)     lkm := 0;
(*7*)     repeat
(*8*)        write('Anna luku, -999 lopettaa: ');
(*9*)        readln(luku);
(*10*)       if luku <> -999 then summa := summa+luku;
(*11*)       if luku <> -999 then lkm := lkm+1
(*12*)    until luku = -999;
(*13*)    if lkm > 0 then writeln('Keskiarvo on  ', summa / lkm)
(*14*) end.
```

(Muuttuja `lkm` on *askeltaja* ja `luku` on *tuoreimman säilyttäjä*.)

Yksisuuntainen lippu

Yksisuuntainen lippu on Boolen muuttuja joka ei saa enää alkuperäistä arvoaan sen jälkeen, kun se on kerran muuttunut. Esimerkkiohjelma tulostaa käyttäjän antamien lukujen summan ja ilmoittaa oliko syötteiden joukossa yhtään negatiivista lukua. *Yksisuuntainen lippu* `neg` tarkkailee (rivillä 10) esiintyykö syötteiden joukossa yhtään negatiivista arvoa ja jos yksikin negatiivinen arvo löytyy, ei muuttuja enää palaa arvoon `false`.

```
(*1*)  program Summa (input,output);
(*2*)  var luku, summa: integer;
(*3*)      neg: Boolean;
(*4*)  begin
(*5*)     summa := 0;
(*6*)     neg := false;
(*7*)     repeat
(*8*)        write('Anna luku, 0 lopettaa: '); readln(luku);
(*9*)        summa := summa + luku;
(*10*)       if luku < 0 then neg := true
(*11*)    until luku = 0;
(*12*)    writeln('Summa on ', summa);
(*13*)    if neg then writeln('Joukossa oli negatiivisia lukuja')
(*14*) end.
```

(Muuttuja `luku` on *tuoreimman säilyttäjä* ja `summa` on *kokooja*.)
Yksisuuntaista lippua voidaan käyttää myös esimerkiksi tarkkailemaan virheen esiintymistä syöttötiedoissa, jotta ohjelma huomaisi pyytää syötteitä uudelleen.

Seuraaja

Seuraaja saa aina arvokseen jonkin tietyn toisen muuttujan vanhan arvon. Esimerkkiohjelma pyytää käyttäjältä 12 kokonaislukua ja kertoo lopuksi, mikä oli suurin kahden perättäisen syötetyn luvun ero. Muuttuja `edellinen` on *seuraaja*: se seuraa muuttujaa `nykyinen` (rivillä 8).

```
(*1*)  program SuurinEro (input,output);
(*2*)  var kuukausi, nykyinen, edellinen, suurinEro: integer;
(*3*)  begin
(*4*)     write('Anna 1. arvo: '); readln(edellinen);
(*5*)     write('Anna 2. arvo: '); readln(nykyinen);
(*6*)     suurinEro := nykyinen - edellinen;
(*7*)     for kuukausi := 3 to 12 do begin
(*8*)        edellinen := nykyinen;
(*9*)        write('Anna ', kuukausi, '. arvo: ');
(*10*)       readln(nykyinen);
(*11*)       if nykyinen - edellinen > suurinEro
(*12*)          then suurinEro := nykyinen - edellinen
(*13*)    end;
(*14*)    writeln('Suurin ero oli ', suurinEro)
(*15*) end.
```

(Muuttuja `kuukausi` on *askeltaja*, `nykyinen` on *tuoreimman säilyttäjä* ja `suurinEro` on *sopivimman säilyttäjä*.)
Seuraajia käytetään paljon linkitettyjen tietorakenteiden yhteydessä osoittamaan käsiteltävää alkiota edeltänyttä alkiota.

Tilapäissäilö

Muuttuja on *tilapäissäilö*, jos sen arvoa tarvitaan aina vain hyvin lyhyen ajan. Esimerkkiohjelma tulostaa syötteenään saamasta kahdesta luvusta ensin suuremman ja sitten pienemmän. Muuttujien sisällöt vaihdetaan keskenään käyttämällä *tilapäissäilönä* muuttujaa `tmp` (riveillä 7-9), jonka arvolla ei ole jatkossa merkitystä (vaikka ohjelma jatkuisi kuinka pitkään tahansa.)

```
(*1*)  program Vaihda (input, output);
(*2*)  var luku1, luku2, tmp: integer;
(*3*)  begin
(*4*)     write('Anna luku: '); readln(luku1);
(*5*)     write('Anna toinen luku: '); readln(luku2);
(*6*)     if luku1 < luku2 then begin
(*7*)        tmp := luku1;
(*8*)        luku1 := luku2;
(*9*)        luku2 := tmp
(*10*)    end;
(*11*)    writeln('Luvuista suurempi on ', luku1,
(*12*)            ' ja pienempi on ', luku2)
(*13*) end.
```

Tilapäissäilöä käytetään usein *järjestelijän* kahden alkion keskinäisen paikan vaihtamiseen.

Järjestelijä

Järjestelijä on taulukko, jota käytetään siinä olevien tietojen uudelleen järjestämiseen sen jälkeen, kun taulukko on ensin alustettu joillakin arvoilla. Esimerkkiohjelma pyytää käyttäjältä merkki kerrallaan yhteensä kymmenen merkkiä *järjestelijään* `merkki`, kääntää niiden järjestyksen taulukossa ja lopuksi tulostaa merkit tässä käännetyssä järjestyksessä.

```
(*1*)  program Kaanna (input, output);
(*2*)  var merkki: array[1..10] of char;
(*3*)      tmp: char;
(*4*)      i: integer;
(*5*)  begin
(*6*)     for i := 1 to 10 do begin
(*7*)        write('Anna kirjain: '); readln(merkki[i])
(*8*)     end;
(*9*)     for i := 1 to 5 do begin
(*10*)       tmp := merkki[i];
(*11*)       merkki[i] := merkki[11-i];
(*12*)       merkki[11-i] := tmp
(*13*)    end;
(*14*)    for i:=1 to 10 do write(merkki[i]);
(*15*)    writeln
(*16*) end.
```

(Muuttuja `tmp` on *tilapäissäilö* ja `i` on *askeltaja*.)
Järjestelijää voidaan käyttää taulukon lajitteluun tai muuhun uudelleenjärjestelyyn.

Muuttujien roolit: Tehtäviä

Mitä muuttujien rooleja näissä ohjelmissa käytetään?

1.

```
program square (input, output);

(* Drawing a square *)

const maxSide = 78;      (* Max length for sides             *)

var character : char;    (* Character to be used for drawing *)
    side,                (* Length of sides                  *)
    i, j      : integer; (* Counters for side lengths        *)

begin

    write('Enter character for drawing: '); readln(character);

    write('Enter side length          : '); readln(side);
    while (side < 1) or (side > maxSide) do begin
        writeln('Length incorrect. Re-enter: '); readln(side)
    end;

    writeln;

    for i := 1 to side do begin
        for j := 1 to side do write(character);
        writeln
    end

end.
```

Vastaus:

Muuttujan nimi	Muuttujan rooli
character	
side	
i	
j	

2.

```
program closest (input, output);

(* Find the closest number *)

var original,           (* Closest to this one  *)
    data,               (* Current input data   *)
    closest: integer;   (* Closest found so far *)

begin

    write('Enter any number: '); readln(original);

    write('Enter a positive number (negative to end): ');
    readln(data);
    closest := data;

    while data >= 0 do begin
        if abs(data-original) < abs(closest-original)
            then closest := data;
        write('Enter a positive number (negative to end): ');
        readln(data);
    end;

    if closest < 0
       then writeln('No positive value entered.')
       else writeln('The closest to ', original, ' was ', closest)
end.
```

Vastaus:

Muuttujan nimi	Muuttujan rooli
original	
data	
closest	

3.

```
    program growth (input, output);

    (* Growth of capital on a bank account *)

    var capital,            (* Capital on the bank account *)
        prevcapital,        (* Capital of the previous year *)
        percent,            (* Interest rate               *)
        interest: real;     (* Interest in current year    *)
        years,              (* Time to consider            *)
        i       : integer;  (* Year counter                *)
        inputOk : Boolean;  (* Is input valid ?            *)

    begin

        prevcapital := 0;
        write('Enter capital (positive or negative): ');
readln(capital);
        inputOk := false;
        while not inputOk do begin
            write('Enter interest rate (%): '); readln(percent);
            write('Enter time (years) : '); readln(years);
            inputOk := (percent > 0) and (years > 0);
            if not inputOk then begin
                writeln;
                writeln('Invalid data. Re-enter.')
            end
        end;
        writeln;

        for i := 1 to years do begin
            interest := capital * percent / 100;
            capital := capital + interest;
            writeln('After ', i : 2, 'years: interest is ',
               interest : 11 : 2,
               ', total capital is ', capital : 12 : 2,
               ' and change of capital is ', capital - prevcapital : 12
: 2);
            prevcapital := capital
        end
    end.
```

Vastaus:

Muuttujan nimi	Muuttujan rooli
capital	
prevcapital	
percent	
interest	
years	
i	
inputOk	

Liite 3

Tehtävä 1

Kirjoita 4 adjektiivia (esim. ujo) kuvasta:

_____
_____
_____
_____

_____
_____
_____
_____

_____
_____
_____
_____

_____
_____
_____
_____

_____
_____
_____
_____

Tehtävä 2

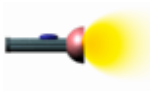
Kirjoita 4 verbiä (esim. juoda) kuvasta:

_____
_____
_____
_____

_____
_____
_____
_____

_____
_____
_____
_____

_____
_____
_____
_____



_____
_____
_____
_____

Liite 4

Tehtävä 1

Kirjoita 4 adjektiivia (esim. ujo) kuvasta:

_____
_____
_____
_____

_____
_____
_____
_____

_____
_____
_____
_____

_____
_____
_____
_____

_____
_____
_____
_____

Tehtävä 2

Kirjoita 4 verbiä (esim. juoda) kuvasta:

_____
_____
_____
_____

_____
_____
_____
_____

_____
_____
_____
_____

_____
_____
_____
_____

_____
_____
_____
_____

Tehtävä 3 Liite 5

PlanAni on ohjelma-animaattori, jonka avulla voidaan esittää muuttujien käyttöä ohjelmissa. Ohjelman suoritus alkaa painamalla Suorita-näppäintä.
PlanAni ilmoittaa etukäteen kuittausruudulla seuraavasta askeleestaan ja ilmoitus pitää kuitata pois painamalla OK:ta tai enteriä ennen kuin ohjelman suoritus jatkuu. Takaisin alkuun pääsee Alkuun-näppäimellä. PlanAnin tulostusalue on alhaalla oleva paperilappu ja syötteet laitetaan sen oikealla puolella olevalle lautaselle.

Suorita animaattorilla näytöllä oleva ohjelma ja merkitse suorituksen aikana muuttujille sijoitetut arvot alla oleville riveille.
Kun ohjelma pyytää syötettä, niin **anna syötteeksi luku 6.**

Mitä arvoja **last_fib** saa ohjelman suorituksen aikana:

_____

Mitä arvoja **fib** saa ohjelman suorituksen aikana:

_____

Mitä arvoja **temp** saa ohjelman suorituksen aikana:

_____

Mitä arvoja **number** saa ohjelman suorituksen aikana:

_____

Mitä arvoja **i** saa ohjelman suorituksen aikana:

_____

Tehtävä 4: a) Mitä roolia kuva esittää? Roolit on listattu alhaalla. Liite 6
b) Arvioi kouluarvosanalla (4-10) kuinka hyvin kuva esittää roolia.
c) Mikä kuva esittäisi roolia kympin arvoisesti? Kirjoita tai piirrä!

	a)	b)	c)
			
			
			
			
			

Muuttujien roolit:

Askeltaja on muuttuja, joka saa kaikki arvonsa alkuarvon asetuksen jälkeen ennustettavissa olevasta sarjasta.
Järjestelijä on taulukko, jota käytetään alustuksen jälkeen vain tietojen järjestämiseen toisella tavalla.
Kiintoarvo on muuttuja, jonka arvo ei alustuksen jälkeen muutu.
Kokooja on muuttuja, joka saa arvonsa summaamalla tai muulla tavalla kumuloimalla käsiteltävänä olevasta sarjasta arvoja.
Seuraaja on muuttuja, joka saa arvokseen toisen muuttujan vanhan arvon.
Sopivimman säilyttäjä on muuttuja, joka säilyttää parhaan tai sopivimman arvon jostakin käsiteltävänä olevasta sarjasta arvoja.
Tilapäissäilö on muuttuja, joka säilyttää arvonsa vain hyvin vähän aikaa.
Tuoreimman säilyttäjä on muuttuja, joka säilyttää viimeisimmän arvon jostakin käsiteltävänä olevasta sarjasta arvoja.
Yksisuuntainen lippu on muuttuja, jolla on kaksi mahdollista arvoa ja joka ei voi saada enää alkuarvoaan muutoksen jälkeen.

Tehtävä 4: a) Mitä roolia kuva esittää? Roolit on listattu alhaalla. Liite 7
b) Arvioi kouluarvosanalla (4-10) kuinka hyvin kuva esittää roolia.
c) Mikä kuva esittäisi roolia kympin arvoisesti? Kirjoita tai piirrä!

	a)	b)	c)
			
			
			
			
			

Muuttujien roolit:

Askeltaja on muuttuja, joka saa kaikki arvonsa alkuarvon asetuksen jälkeen ennustettavissa olevasta sarjasta.
Järjestelijä on taulukko, jota käytetään alustuksen jälkeen vain tietojen järjestämiseen toisella tavalla.
Kiintoarvo on muuttuja, jonka arvo ei alustuksen jälkeen muutu.
Kokooja on muuttuja, joka saa arvonsa summaamalla tai muulla tavalla kumuloimalla käsiteltävänä olevasta sarjasta arvoja.
Seuraaja on muuttuja, joka saa arvokseen toisen muuttujan vanhan arvon.
Sopivimman säilyttäjä on muuttuja, joka säilyttää parhaan tai sopivimman arvon jostakin käsiteltävänä olevasta sarjasta arvoja.
Tilapäissäilö on muuttuja, joka säilyttää arvonsa vain hyvin vähän aikaa.
Tuoreimman säilyttäjä on muuttuja, joka säilyttää viimeisimmän arvon jostakin käsiteltävänä olevasta sarjasta arvoja.
Yksisuuntainen lippu on muuttuja, jolla on kaksi mahdollista arvoa ja joka ei voi saada enää alkuarvoaan muutoksen jälkeen.

Tehtävä 5: a) Miten hyödyllistä roolien esittäminen kuvilla mielestäsi on?
Vastaa kouluarvosanalla 4-10: _____

b) Mitä apua siitä voisi olla?

_____

_____

_____

_____

_____

c) Mitä haittaa siitä voisi olla?

_____

_____

_____

_____

_____

Tehtävä 6: Mitä muuttujien rooleja näissä ohjelmissa käytetään? Liite 9

a)

```
program diceGame (input, output);

(* Simulate a dice game. *)

var die1, die2,              (* Values of dice thrown          *)
    total1, total2: integer; (* Sums of values of each player  *)
    firstPlayer: boolean;    (* First player's turn ?          *)

begin

    Randomize;
    total1 := 0;
    total2 := 0;
    firstPlayer := True;

    while (total1 < 100) and (total2 < 100) do begin
        die1 := Random(6) + 1;
        die2 := Random(6) + 1;
        if firstPlayer then write('First') else write('Second');
        writeln(' player throws: ', die1, die2);

        if firstPlayer then
            total1 := total1 + die1 + die2
        else
            total2 := total2 + die1 + die2;

        firstPlayer := not firstPlayer;
    end;

    write('Player ');
    if total1 > total2 then write('First') else write('Second');
    write(' player won the game.')

end.
```

Vastaus:

Muuttujan nimi	Muuttujan rooli
die1	
die2	
total1	
total2	
firstPlayer	

Roolit: askeltaja, järjestelijä, kiintoarvo, kokooja, seuraaja, sopivimman säilyttäjä, tilapäissäilö, tuoreimman säilyttäjä, yksisuuntainen lippu.

b)

```
program histogram (input, output);

(* Draw a histogram *)

const longest = 40;  (* Longest bar *)

var amount : array [1..12] of real; (* Data for drawing     *)
    max : real;                     (* Maximum data element *)
    month,                          (* Current month        *)
    monthCount,                     (* Number of months     *)
    i : integer;

begin

    write('Enter the number of months (1-12): ');
    readln(monthCount);

    for month := 1 to monthCount do begin
        write('Enter amount for month ', month : 2, ': ');
        readln(amount[month]);
        if i = 1 then max := amount[1]
        else if max < amount[month] then max := amount[month]
    end;
    writeln;

    for month := 1 to monthCount do begin
        write(month : 2, ': ');
        for i := 1 to round(amount[month] / max * longest)
            do write('*');
        writeln
    end

end.
```

Vastaus:

Muuttujan nimi	Muuttujan rooli
amount	
max	
month	
monthCount	
i	

Roolit: askeltaja, järjestelijä, kiintoarvo, kokooja, seuraaja, sopivimman säilyttäjä, tilapäissäilö, tuoreimman säilyttäjä, yksisuuntainen lippu.

c)

```
program smoothedAverage (input, output);

(* Largest average of three consecutive months *)

var month: integer;               (* Current month            *)
    current, previous, preceding, (* Data for three months    *)
    average,                      (* Current average          *)
    largest: real;                (* Largest one found so far *)

begin

    write('Enter 1. value: '); readln(preceding);
    write('Enter 2. value: '); readln(previous);
    write('Enter 3. value: '); readln(current);
    largest := (current + previous + preceding) / 3;

    for month := 4 to 12 do begin
        preceding := previous;
        previous := current;
        write('Enter ', month, '. value: '); readln(current);
        average := (current + previous + preceding) / 3;
        if average > largest then largest := average
    end;

    writeln('Largest three month average was ', largest)

end.
```

Vastaus:

Muuttujan nimi	Muuttujan rooli
Month	
Current	
Previous	
Preceding	
Average	
Largest	

Roolit: askeltaja, järjestelijä, kiintoarvo, kokooja, seuraaja, sopivimman säilyttäjä, tilapäissäilö, tuoreimman säilyttäjä, yksisuuntainen lippu.

Kokeen ohjeet: ensimmäinen tehtävänippu Liite 10

Tervetuloa osallistumaan kokeeseen, joka on osa tekeillä olevaa graduani. Olen todella kiitollinen avustanne!

Kokeessa tutkitaan eräiden apuvälineiden sopivuutta ihmisten ajattelulle. Kokeessa ei arvioida teidän osaamistanne tai kykyjänne vaan tutkittavien välineiden sopivuutta tavallisille ihmisille. Kokeen tuloksia käsitellään luottamuksellisina eikä yksittäisen henkilön tietoja julkaista missään. Tuloksia tullaan esittelemään vain yhtenä kokonaisuutena. Nimet laitetaan tehtäväpapereihin vain siitä syystä, että saadaan kerättyä saman henkilön vastaukset kokonaisuudeksi.

Koetehtävät annetaan paperilla kahdessa erässä. Annan suullisesti ohjeita aina ennen tehtävien tekoa. Kuunnelkaa ohjeita! Tehtävät tehdään siinä järjestyksessä ja aikataulussa kuin ohjaaja sanoo. Jos on jotain kysymistä kokeen aikana, nostakaa käsi ylös, niin tulen paikalle. Olisiko jo nyt jotain kysymistä?

Ensimmäinen tehtävänippu sisältää kansilehden lisäksi 6 sivua luettavaa tekstiä ja lopuksi tekstiin perustuvia tehtäviä 3 kpl. Tehtäviä tehdessä voi vapaasti käyttää hyväksi kaikkia paperinipun sivuja. Muistakaa laittaa nimenne sille varattuun kohtaan.

(Pirkko jakaa tehtäväpaperit kaikille).
Aikaa on 15 minuuttia.

Nyt loppuu aika. Pyydän teitä palauttamaan paperit minulle ja poistumaan luokasta tauolle 15 minuutiksi. Sitten koe jatkuu uusilla tehtävillä täällä samassa paikassa.

Tässä välissä tarkistetaan Pirkon kanssa vastaukset ja laitetaan yli 7 oikein saaneet omaan pinoonsa ja 1-7 oikein saaneet omaan pinoonsa. Tehdään kaksi ryhmää siten, että laitetaan niihin osallistujia tasaisesti kummastakin pinosta. Pirkko saa toisen ryhmän paperit ja nimet ja minä toisen.

Kokeen ohjeet: toinen tehtävänippu

Tervetuloa takaisin, koe jatkuu. Pyydän teitä jakautumaan kahteen ryhmään nimenhuudon perusteella.

(Pirkko huutaa nimiä kontrolliryhmään. Loput ovat ja minun rooliryhmäläisiä. Pirkko vie kontrolliryhmäläiset toiseen mikroluokkaan, rooliryhmä jää paikan päälle.)

(Tästä eteenpäin Pirkko ja minä johdetaan koetta ja tässä meidän sanomiset:)
Nyt pyydän teitä valitsemaan paikat itsellenne eturivistä.

Toinen tehtävänippu sisältää kansilehden lisäksi 5 tehtäväsivua. Tehtävät tehdään siinä järjestyksessä ja aikataulussa kuin ohjaaja sanoo. Sivua saa kääntää vain silloin kuin ohjaaja sanoo.

Tämä on erittäin tärkeää kokeen onnistumisen kannalta: myöhempänä olevia tehtäviä tai jo tehtyjä tehtäviä ei saa kurkkia eikä muuttaa.

(Kokeenjohtaja jakaa tehtävät osallistujille.)

Kirjoittakaa nimenne sille varattuun paikkaan kansilehdelle.

Nyt saatte kääntää esille tehtävän 1 ja aloittaa sen tekemisen. Aikaa on 5 minuuttia.

Aika loppuu.
Nyt saatte kääntää esille tehtävän 2.
Onhan kaikilla edessään nyt tehtävä 2 ?
Selvä, voitte aloittaa sen tekemisen. Aikaa on 5 minuuttia.

Aika loppuu.
Nyt saatte kääntää esille tehtävän 3. Tämän tehtävän tekemiseen tarvitsette myös tietokonetta ja siihen asennettua PlanAni-ohjelmaa. Pyydän teitä nyt laittamaan näyttöihin virran päälle. PlanAni on koneillanne valmiina käyttöä varten. Jos näytönsäästäjä on päällä, painakaa jotain nappulaa, niin saatte PlanAnin näkyviin. Tehtäväpaperissa on lyhyt opastus PlanAnin käyttöön. Lukekaa tehtävät huolellisesti ennen kuin alatte suorittaa ohjelmaa PlanAnilla.
Onhan kaikilla nyt edessään tehtävä 3 ja näytöllä PlanAni näkyvissä?
Selvä, voitte aloittaa sen tekemisen. Aikaa on 10 minuuttia.

Aika loppuu.
Pyydän teitä laittamaan virran pois näytöstä.
(Ohjaaja kiertää luokassa katsomassa, että kaikilta on virta pois näytöstä.)
Nyt voitte kääntää esille tehtävän 4. Aikaa tämän tehtävän tekemiseen on 10 minuuttia.

Aika loppuu.
Nyt saatte kääntää esille tehtävän 5 ja aloittaa sen tekemisen. Aikaa on 10 minuuttia.

Aika loppuu.
Nyt saatte kääntää esille tehtävän 6 ja aloittaa sen tekemisen. Aikaa on 10 minuuttia.

Aika loppuu.
Nyt on kaikki tehtävät tehty. Kiitän teitä osallistumisestanne! Voisitteko nyt palauttaa tehtävät minulle. Annan tässä samalla nämä hyvin ansaitut lounasliput teille. Niiden arvo on maksimissaan 6 euroa ja ne käyvät alakerran Amican ruokalaan.
Pyydän, että ette kerro ulkopuolisille tietoja kokeen yksityiskohdista, jotta myöhemmin kokeeseen tulevien henkilöiden suoritukset eivät vaarantuisi.