Muuttujien roolit ohjelmistoasiantuntijoiden kirjoittamissa Java-ohjelmissa

Kasper Heikkilä

06.06.2008

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

Tiivistelmä

Lähes kaikkiin ohjelmointiparadigmoihin kuten proseduraaliseen, funktionaaliseen ja oliopohjaiseen paradigmaan perustuvissa ohjelmissa muuttujien esiintymiseen voidaan liittää tietyt roolit käyttämällä pientä ryhmää stereotyyppisiä malleja, joita kutsutaan muuttujien rooleiksi. Tutkielmassa selvitetään, kuinka nämä roolit soveltuvat Java-kielellä kirjoitettuihin olio-ohjelmiin, mikä on kunkin roolin esiintymistiheys, ja olisiko olemassaoleviin muuttujien rooleihin tehtävä lisäyksiä, kun kyseessä ovat ohjelmistoasiantuntijoiden kirjoittamat Java-ohjelmat. Tutkielmassa tarkastellaan kolmea erilaista ohjelmaa, joiden muuttujat on roolitettu tutkimuksessa määritetyillä olioiden käsittelyyn kohdistuvilla erikoissäännöillä sekä aikaisemmissa tutkimuksissa luotujen muuttujien yleisten roolitussääntöjen perusteella. Saatuja tuloksia verrataan myös aikaisemmin roolitettuihin aloitteleville ohjelmoijille suunnatuista mutta ohjelmistosiantuntijoiden kirjoittamista ohjelmista saatuihin tuloksiin. Ohjelmistoasiantuntijoiden kirjoittamien Java-ohjelmien kesken ei muuttujien roolien jakaumissa ollut tilastollisesti merkitseviä eroja. Roolien jakaumat erosivat toisistaan tilastollisesti merkitsevästi ohjelmistoasiantuntijoiden aloitteleville ohjelmoijille suunnattujen ohjelmien välillä sekä näiden ja ohjelmistoasiantuntijoiden kirjoittamien Java-ohjelmien välillä. Tulosten perusteella nykyisin käytössäoleva muuttujien roolitusmalli soveltuu myös ohjelmistoasiantuntijoiden kirjoittamiin Java-ohjelmiin, eikä rooleihin ole tarvetta tehdä lisäyksiä.

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

Avainsanat: Java, muuttujien roolit, ohjelmistoasiantuntijat, olio-ohjelmointi

Abstract

Practically in all programs based on programming paradigms, such as procedural, functional and object-oriented programming, a small group of stereotypical models called roles of variables can be attached to the occurrence of variables. The present thesis investigates how these roles apply to object-oriented programs written in Java, what is the occurrence of each role and if the existing roles should be supplemented with additional ones when examining Java programs written by expert level programmers. The thesis examined three different programs, the variables of which were classified with a special set of rules determined in the present study and with general classification guidelines set by previous research. The data are also compared to earlier classification results obtained from programs written by experts but targeted at novice level programmers. The distribution of the roles was significantly different between programs written by experts to novices. The distribution was also different between the programs written for novices and the Java programs written by experts, while the differences between the three Java programs were nonsignificant. The results show that the present model of role classification can be also applied to Java programs written by experts and there is no urgent need to supplement the roles currently in use.

ACM-classes (ACM Computing Classification System, 1998 version): D.m, K.3.2

Key words: expert programmers, Java, object-oriented programming, roles of variables

Sisältö

1 Johdanto

Ohjelmoinnissa on edetty pitkä matka puhtaiden konekielten käytöstä, jotka olivat ainoa tapa ohjelmoida 1940-luvulta aina 1950-luvun alkuun (Davis, 2003; Sebesta, 2003). Symboliset konekielet (assembler languages), jotka vapauttivat ohjelmoijat muun muassa muistiosoitteiden vaivalloisesta laskennasta, kehitettiin 1950-luvulla. Nykyisin laajimmin käytössä ovat 1980-luvulta lähtien kehitetyt kolmannen sukupolven korkean tason lausekielet, joihin kuuluu vuonna 1995 julkaistu, olioiden käyttöön perustuva Java.

Olioihin perustuvien korkean tason lausekielten opetuksessa on suositeltu käytettäväksi kahden askeleen ohjelmaa. Ensimmäisessä vaiheessa ohjelmoinnin perusteiden matemaattisten funktioiden opetus suoritetaan jollain funktionaalisella ohjelmointikielellä kuten Lispillä (Felleisen *et al.*, 2004). Toisessa vaiheessa siirrytään proseduraalisten kielten kuten C:n kautta olio-ohjelmoinnin pariin käyttäen käsitettä luokka. Tämä mahdollistaa sen, että oppilaat oppivat ymmärtämään olioiden rakenteet ilman niihin liittyviä vaikeita yksityiskohtia. Suositukset ja todellisuus eivät kuitenkaan ole aina kohdanneet, ja sekä luokkien että matemaattisten funktioiden käytössä ja opetuksessa on ollut ongelmana niiden staattinen luonne (Sajaniemi *et al.*, 2006). Ongelmaa on lähdetty ratkomaan muuttujien roolit -käsitteellä, joka esittää ohjelmoinnin dynaamisen puolen helposti omaksuttavassa muodossa (Sajaniemi, 2002a).

Muuttujien roolit pysyvät samoina oli kyseessä ohjelmoinnin opetus ja opetuksessa käytetyt noviiseille suunnatut yksinkertaiset ohjelmat tai ohjelmistoasiantuntijoiden kirjoittamat ohjelmat. Tutkittaessa rooleja erot aloittelevien ohjelmoijien ja ohjelmistoasiantuntijoiden välillä ovat pääasiassa vain käytettyjen muuttujien roolien jakaumassa ja roolien ymmärtämisessä, itse roolit pysyvät muuttumattomina (Sajaniemi, 2002b; Sajaniemi & Navarro Prieto, 2005). Tutkimukseni perustuu ohjelmistoasiantuntijoiden kirjoittamien olioparadigmaan perustuvien ohjelmien muuttujien roolien analysoimiseen sekä siihen, kuinka löydökset mahdollisesti vaikuttavat muuttujien roolien luokitteluun ja ehkä myöhemmin myös olio-ohjelmoinnin opetukseen.

Tutkielmani aineistona on kolme ohjelmistoasiantuntijoiden kirjoittamaa Java-ohjelmaa, kolme aikaisemmin analysoitua Pascal-kielen oppikirjaa (Sajaniemi, 2002b), kaksi aikaisemmin analysoitua Java-kielen oppikirjaa (Byckling *et al.*, 2005) sekä neljä aikaisemmin analysoitua ML-kielen oppikirjaa (Kulikova, 2005). Ohjelmis-

toasiantuntijoiden kirjoittamista Java-ohjelmista roolitan kustakin 150 muuttujaa sekä ensimmäisenä tutkitusta Megamek-ohjelmasta roolitan lisäksi 250 parametriä. Tekemiäni roolituksia vertaan keskenään χ^2-testillä, myös aikaisemmista roolituksista saadut tulokset käsitellään samoin. Ryhmien sisällä saatujen tulosten lisäksi vertaan eri ryhmiä myös keskenään. Koska Java- ja Pascal-oppikirjoista tehdyt roolitukset perustuvat vanhempiin version 1.1 mukaisiin roolitussääntöihin ja koska ohjelmistoasiantuntijoiden kirjoittamien ohjelmien roolitukset perustuvat version 1.1 kanssa lähes identtisiin version 1.2 mukaisiin roolitussääntöihin, uusien roolien toimivuutta testatakseni olen vielä lopuksi muuttanut kaikki nämä tulokset roolitussääntöjen version 2.0 mukaisiksi. Näistä säännöistä poikkeavalla tavalla roolitettuja ML-oppikirjoista otettuja ohjelmia en muuntanut uusimpien roolitussääntöjen mukaisiksi.

Aluksi tarkastelen, kuinka käsite ”muuttujien roolit” on syntynyt ja tarkentunut nykyiseen muotoonsa, ja mitä asioita tulee ottaa huomioon muuttujien roolien käsittelyssä, kun kyseessä on olio-ohjelmointi. Aineisto ja tutkimusmenetelmät -luvussa käyn läpi tutkimukseni taustoja ja päämääriä, mitä löydöksiä aikaisemmissa tutkimuksissa on tehty, ja kuinka nämä vaikuttivat oman tutkimukseni tekemiseen. Lisäksi käsittelen lyhyesti, kuinka aineisto valittiin, läpikäytiin ja roolitettiin sekä kuinka materiaali analysoitiin tilastollisin menetelmin. Luvussa 4 käsittelen saadut tulokset ohjelmittain ja ohjelmatyypeittäin. Luku 5 on tulosten tarkastelua ja tutkimustulosten tuomien vastausten pohdintaa aihepiirin sisällä sekä näiden kelpoisuuden pohdintaa. Yhteenveto luo yleisen katsauksen tutkimukseen ja tutkimustuloksiin.

2 Roolit ja niihin liittyvä aiempi tutkimus

Roolit ja niihin liittyvä aiempi tutkimus -luvussa käsitellään kohdassa 2.1 muuttujien roolien käsitettä eli sitä, mitä muuttujien rooleilla yleensä tarkoitetaan. Kohdassa 2.2 käsitellään lyhyesti muuttujien rooleja, kun kyseessä ovat olio-ohjelmat, ja kohdassa 2.3 on lyhyehkö katsaus aihepiirin aikaisempiin tutkimuksiin.

2.1 Muuttujien roolit käsitteenä

Muuttujien roolit ovat osa ohjelmistoasiantuntijoiden mentaalista representaatiota (Sajaniemi & Navarro Prieto, 2005), millä tarkoitetaan tapoja käsitellä, järjestää ja varastoida tietoja aivoissa (Johnson-Laird, 1987). Muuttujien roolit ovat vain pieni tarkoin määrätty osa tästä mentaalisesta representaatiosta, joten muuttujien roolien opettaminen on mahdollista niin noviiseille kuin ohjelmistoasiantuntijoillekin. Muuttujien roolien yleismaailmallisen luonteen ansiosta niitä on mahdollista käyttää hyödyksi sekä opetuksessa että tutkimuksessa. Tässä lähestymistavassa muuttujan rooli kuvaa muuttujan dynaamista luonnetta sen käyttäytymisen pohjalta: kuinka muuttuja saa arvonsa toisten muuttujien, vakioiden ja ulkoisten tapahtumien kuten käyttäjän syötteen johdosta (Sajaniemi *et al.*, 2006). Muuttujan käyttötapa ei vaikuta muuttujan rooliin, eli vakiotyyppinen muuttuja on *kiintoarvo* riippumatta siitä, käytetäänkö sitä yksittäisessä jakolaskussa jakajana vai rajoittamassa silmukan toistokertoja (kuva 1). Lisäksi roolit liittyvät käyttäytymistä kuvaavan luonteensa johdosta enemmän ohjelmien osien välisiin loogisiin rakenteisiin eli syvärakenteisiin (Détienne, 2002) kuin pintarakenteisiin eli siihen, mitä sijoituslauseita muuttujien päivittämisessä käytetään.

Kuvassa 1 on esimerkki muuttujien erilaisista rooleista. Parametri (ylaraja) kuten myös muuttuja (j) ovat rooliltaan *kiintoarvoja*, koska niiden arvot eivät muutu alustamisen jälkeen. Muuttuja (j) on rooliltaan *askeltaja*, koska se käy läpi sarjan lukuja, joiden arvot voidaan ennakoida heti, kun sarjan suoritus alkaa. Muuttuja (summa) on rooliltaan *kokooja*, koska siihen kootaan muuttujan (i) eri arvojen summa.

Muuttujien roolien perusjoukoksi otettiin alun perin Greenin & Cornahin (1984) Programmer's Torch -nimisen muuttujien roolien etsimiseen tarkoitetun työkaluehdotelman yhteydessä esitellyt roolit (taulukko 1; Sajaniemi, 2002a). Tämän perusjoukon ja aikaisempien muuttujien käyttöä koskevien tutkimusten pohjalta (Ehrlich & Soloway,

```
public static int SummaDivYlaraja(int ylaraja) {
  int j,i,summa;

  j = ylaraja;

  for (summa=0, i=1; i<=j; i++)
    summa += i;

  summa = summa/j;
  return summa;
}
```

Kuva 1: Esimerkki, jossa muuttujan (j) rooli on *kiintoarvo* käyttötavasta riippumatta, muuttuja (i) on *askeltaja*, muuttuja (summa) on *kokooja*, ja parametri (ylaraja) on *kiintoarvo*.

1984; Rist, 1989) tehtiin ensimmäinen versio 1.0 muuttujien rooleista (taulukko 2; Sajaniemi, 2002a). Tätä tarkennettiin myöhempien tutkimusten perusteella lisäämällä uusi rooli *muuntaja*, jolloin päästiin version 1.1 mukaisiin rooleihin (taulukko 3; Ben-Ari & Sajaniemi, 2004), jotka ovat pieniä tarkennuksia lukuunottamatta identtiset verrattuna version 1.2 mukaisiin rooleihin (taulukko 4), joita käytin analysoidessani ohjelmistoasiantuntijoiden kirjoittamia Java-ohjelmia. Niihin rooleihin, joita käytin tutkiessani uusimpien roolien soveltuvuutta eri paradigmoissa päästiin lisäämällä joukkoon vielä *säiliö* ja *kulkija* (taulukko 5; Sajaniemi *et al.*, 2006). Tutkimuksessa *muuntaja*-roolin huomattiin voivan sisältyä *väliaikais*-rooliin ja *tuoreimman säilyttäjä* -rooliin (Sajaniemi *et al.*, 2006), mutta tutkielmassani tehdyssä ohjelmistoasiantuntijoiden kirjoittamien Java-ohjelmien muuttujien roolituksessa tätä muutosta ei alunperin huomioitu, vaikka roolit on loppuvaiheessa muunnettu vastaamaan uusimpia roolitussääntöjä. Kulikovan (2005) ML-ohjelmien rooleja käsittelevässä tutkimuksessa käyttämää *modifier*-roolia (mdf), joka on vain ML-kielisille ohjelmille tyypillinen listojen muokkauksessa käytettävä rooli, ei huomioitu tutkielmani tuloksissa. Toinen vain ML-kielisten ohjelmien roolituksessa käytetty rooli *selector* (sel) käsiteltiin tutkielmassani *kiintoarvo*-roolina (fix). *Selector*-rooli on funktion paluuarvo, joka saa funktion eri kutsukerroilla mahdollisesti eri arvon, mutta jonka rooli tästä huolimatta on *kiintoarvo*.

Taulukko 1: Muuttujien roolit Green & Cornah (1984) mukaan.

Rooli	*Kuvaus*
Kiintoarvo (Constant)	Muuttuja, jonka arvo ei muutu alustamisen jälkeen
Askeltaja (Counter)	Muuttuja, joka käy läpi arvoja jollain systemaattisella ja ennustettavalla tavalla
Silmukan askeltaja (Loop counter)	Muuttuja, jota käytetään kontrolloimaan silmukan suorittamiskertoja
Tuoreimman säilyttäjä (”Most-recent” holder)	Muuttuja, jossa on viimeisin jostakin joukosta läpikäyty arvo tai yksinkertaisesti vain arvo, joka on syötetty viimeksi
Sopivimman säilyttäjä (”Best of” holder)	Muuttuja, jossa on ”paras” tai jollain muulla tavalla halutuin arvo tähän asti läpikäydyistä arvoista
Kontrolli-muuttuja (Control variable)	Muuttuja, joka kontrolloi ohjelmakoodin suoritusta ohjelmassa aikaisemmin olleen tilanteen pohjalta
Alirutiini-muuttuja (Subroutine variable)	Muuttuja, jota käytetään funktioissa kutsujan ja kutsuttavan väliseen kommunikaatioon; parametri, tulos, muuttujaparametri tai paikallinen muuttuja

Taulukko 2: Muuttujien roolit version 1.0 mukaan (Sajaniemi, 2002a).

Rooli	*Kuvaus*
Kiintoarvo	Muuttuja, jonka arvo ei muutu alustamisen jälkeen
Askeltaja	Muuttuja, joka käy läpi arvoja jollain systemaattisella ja ennustettavalla tavalla
Tuoreimman säilyttäjä	Muuttuja, jossa on viimeisin jostakin joukosta läpikäyty arvo tai yksinkertaisesti vain arvo, joka on syötetty viimeksi
Sopivimman säilyttäjä	Muuttuja, jossa on ”paras” tai jollain muulla tavalla halutuin arvo tähän asti läpikäydyistä arvoista
Kokooja	Muuttuja, jonka arvo kerääntyy kaikista tähän mennessä läpikäydyistä arvoista
Seuraaja	Muuttuja, joka saa aina arvokseen toisen tietoalkion vanhan arvon
Yksisuuntainen lippu	Kaksiarvoinen muuttuja, joka ei saa enää alkuperäistä arvoaan sen jälkeen, kun se on kerran muuttunut
Tilapäissäilö	Muuttuja, jonka arvoa tarvitaan vain hyvin lyhyen ajan
Järjestelijä	Tietorakenne, jossa säilytettävät alkiot voidaan järjestellä uudelleen
Muut	Kaikki muut edellisiin ryhmiin sopimattomat

Taulukko 3: Muuttujien roolit version 1.1 mukaan (Ben-Ari & Sajaniemi, 2004).

Rooli	*Kuvaus*
Kiintoarvo	Muuttuja, jonka arvo ei muutu alustamisen jälkeen
Askeltaja	Muuttuja, joka käy läpi arvoja jollain systemaattisella ja ennustettavalla tavalla
Tuoreimman säilyttäjä	Muuttuja, jossa on viimeisin jostakin joukosta läpikäyty arvo tai yksinkertaisesti vain arvo, joka on syötetty viimeksi
Sopivimman säilyttäjä	Muuttuja, jossa on ”paras” tai jollain muulla tavalla halutuin arvo tähän asti läpikäydyistä arvoista
Kokooja	Muuttuja, jonka arvo kerääntyy kaikista tähän mennessä läpikäydyistä arvoista
Muuntaja	Muuttuja, joka saa uuden arvonsa aina samalla laskentatavalla yhden tai useamman muun muuttujan arvoista
Seuraaja	Muuttuja, joka saa aina arvokseen toisen tietoalkion vanhan arvon
Yksisuuntainen lippu	Kaksiarvoinen muuttuja, joka ei saa enää alkuperäistä arvoaan sen jälkeen, kun se on kerran muuttunut
Tilapäissäilö	Muuttuja, jonka arvoa tarvitaan vain hyvin lyhyen ajan
Järjestelijä	Tietorakenne, jossa säilytettävät alkiot voidaan järjestellä uudelleen

Taulukko 4: Muuttujien roolituksessa käytetyt version 1.2 mukaiset tarkennetut roolit (Sajaniemi, 2004).

Rooli	*Alarooli*	*Kuvaus*
fix	?	fixed value
fix	in	obtained through input; no correction
fix	ic	obtained through input; correction
fix	co	assigned using a single constant
fix	va	assigned using a single variable
fix	pa	passed as a parameter
fix	dv	assigned by creating a new dynamic variable
fix	ot	other
org	?	organizer
org	so	used for sorting
org	re	used for reversing
org	ra	used to generate random order
org	ot	other
stp	?	stepper
stp	cn	counter: increase by 1; no known limit
stp	i1	increase by 1; limit known
stp	in	increase by something else
stp	nc	negative counter: decrease by 1; no known limit
stp	d1	decrease by 1; limit known
stp	dn	decrease by something else
stp	a2	alternate between two values
stp	an	alternate between several values; 1,2,...,n,1,2,...
stp	il	stepping through a linked list
stp	ot	other
mrh	?	most-recent holder
mrh	in	obtained through input; no correction
mrh	ic	obtained through input; correction
mrh	ae	obtained from an any data structure element
mrh	dv	obtained by creating a new dynamic variable
mrh	ot	other

Taulukko 4: Muuttujien roolituksessa käytetyt version 1.2 mukaiset tarkennetut roolit (Sajaniemi, 2004) (jatk.).

Rooli	*Alarooli*	*Kuvaus*
gat	?	gatherer
gat	su	sum
gat	de	decrease by subtraction
gat	mu	multiplication
gat	di	distance, e.g., time = curr - time
gat	ot	other
mwh	?	most-wanted holder
mwh	ma	maximum
mwh	mi	minimum
mwh	cl	closest
mwh	fa	farthest away
mwh	fi	first (in a list)
mwh	la	last (in a list)
mwh	ot	other
one	?	one-way flag
one	r1	raise (false -> true); once
one	rn	raise (false -> true); multiple episodes
one	l1	lower (true -> false); once
one	ln	lower (true -> false); multiple episodes
one	ot	other
trn	?	transformation
trn	fc	fixed transformation; for clarity (used only once)
trn	rc	repeated transformation; for clarity
trn	fs	fixed transformation; for statement economy
trn	rs	repeated transformation; for statement economy
trn	fe	fixed transformation; for efficiency (used more than once)
trn	re	repeated transformation; for efficiency
trn	ot	other
fol	?	follower
fol	r1	strictly following

Taulukko 4: Muuttujien roolituksessa käytetyt version 1.2 mukaiset tarkennetut roolit (Sajaniemi, 2004) (jatk.).

Rooli	*Alarooli*	*Kuvaus*
fol	rn	non-strict, e.g., low := middle + 1
fol	ot	other
tmp	?	temporary
tmp	sw	swap
tmp	ot	other
oth	?	other
oth	sc	scalar value
oth	ar	array
oth	ot	should not be possible

Taulukko 5: Muuttujien roolit version 2.0 mukaan (Sajaniemi *et al.*, 2006).

Rooli	*Kuvaus*
Kiintoarvo	Tietoalkio, jonka arvo ei muutu alustamisen jälkeen
Askeltaja	Tietoalkio, joka käy läpi arvoja jollain systemaattisella ja ennustettavalla tavalla
Tuoreimman säilyttäjä	Tietoalkio, jossa on viimeisin jostakin joukosta läpikäyty arvo tai yksinkertaisesti vain arvo, joka on syötetty viimeksi
Sopivimman säilyttäjä	Tietoalkio, jossa on ”paras” tai jollain muulla tavalla halutuin arvo tähän asti läpikäydyistä arvoista
Kokooja	Tietoalkio, jonka arvo kerääntyy kaikista tähän mennessä läpikäydyistä arvoista
Seuraaja	Tietoalkio, joka saa aina arvokseen toisen tietoalkion vanhan arvon
Yksisuuntainen lippu	Kaksiarvoinen tietoalkio, joka ei saa enää alkuperäistä arvoaan sen jälkeen, kun se on kerran muuttunut
Tilapäissäilö	Tietoalkio, jonka arvoa tarvitaan vain hyvin lyhyen ajan
Järjestelijä	Tietorakenne, jossa säilytettävät alkiot voidaan järjestellä uudelleen
Säiliö	Tietorakenne, johon voidaan lisätä ja josta voidaan poistaa alkioita
Kulkija	Tietoalkio, joka käy läpi tietorakennetta

2.2 Muuttujien roolit olio-ohjelmissa

Olio-ohjelmissa kuten Javassa voidaan roolittaa muuttujien ja parametrien lisäksi myös luokkien attribuutteja. Tietyt luokat (oliot) kuten Java-Float (liukuluku; kuva 2) tai Java-String (merkkijono; kuva 3) kapsuloivat tiedon ja funktion yhdeksi kokonaisuudeksi, joka käyttäytyy kuin jokin alkeistyyppinen muuttuja. Esimerkiksi kun Java-Float-olio luodaan ohjelman suorituksen aikana, suoritetaan samalla Java-luokassa Float Float-funktiolla float-arvon asettaminen (kuva 2). Näissä tapauksissa olion roolia käsitellään ainoan attribuuttinsa roolina. Siten esimerkiksi kuvassa 2 esiintyvän muuttujan ”liukuluku” rooliksi tulisi *kiintoarvo*, ja kuvassa 3 esiintyvän muuttujan ”str” rooliksi tulisi myös *kiintoarvo*.

```
// Luodaan uusi Java ’’Float’’ olio:
Float liukuluku = new Float(2.3f);
// Osa Javan ’’Float’’ toteutuksesta,
// joka suorittaa ylläolevan olion luonnin:
public final class Float extends Number implements Comparable {
  private final float value;

  public Float(float value) {
    this.value = value;
  }
}
```

Kuva 2: Esimerkki tiedon ja funktion kapsuloinnista, kun kyseessä on Java-Float.

2.3 Katsaus aikaisempiin tutkimuksiin

Aikaisempien tutkimusten perusteella ohjelmoijien käsitys muuttujien rooleista vaihtelee (Sajaniemi & Navarro Prieto, 2005). Muuttujien käyttäytymisessä voi olla eroja kahden tarkkailijan havaintojen perusteella, vaikka he tutkisivat täsmälleen samaa muuttujaa ja sen identtistä arvojen vaihtumista samoissa sijoituslauseissa (Sajaniemi *et al.*, 2006). Sajaniemi & Navarro Prieton (2005) mukaan ohjelmistoasiantuntijoiden välillä on kaksi variaatiota muuttujien roolituksen identifioinnissa: muuttujien käyttäytymisen luonne elinaikanaan ja erot muuttujien mieltämisessä. Muuttujan arvon toistuva

```
String str = "abc";

// Ylläolevan suorittaminen vastaa:
char data[] = {’a’, ’b’, ’c’};
String str = new String(data);

// Osa Java String-toteutuksesta,
// jolla ylläolevan olion luonti suoritetaan:
public final class String implements
Serializable, Comparable, CharSequence {
 final char[] value;
 final int count;
 final int offset;

 String(char[] data, int offset, int count, boolean dont_copy) {
  if (offset < 0 || count < 0 || offset + count > data.length)
   throw new StringIndexOutOfBoundsException();
  if (dont_copy) {
   value = data;
   this.offset = offset;
  }
  else {
   value = new char[count];
   VMSystem.arraycopy(data, offset, value, 0, count);
   this.offset = 0;
  }
   this.count = count;
 }

 public String(char[] data) {
  this(data, 0, data.length, false);
 }
}
```

Kuva 3: Esimerkki tiedon ja funktion kapsuloinnista, kun kyseessä on Java-String.

jakaminen tai kasvattaminen vakiolla ovat esimerkkejä samankaltaisesta käyttäytymisestä. Osa ohjelmistoasiantuntijoista oli epävarmoja näiden kahden tapauksen samankaltaisuudesta, toiset pitivät niitä samankaltaisena toimintana ja roolittivat ne nykyisin käytössäolevan *askeltaja*-roolin kaltaiseksi, kolmas ryhmä katsoi muuttujien kuuluvan kahteen eri käyttäytymismalliin. Esimerkiksi muuttujien käyttäytymisen luonteesta elinaikanaan voidaan ottaa Fibonaccin lukujono, jossa lasketaan kaksi edellistä lukua yhteen ja saadaan näin kolmannen lukuarvo. Matemaatikko näkee sekvenssin yhtä selkeästi kuin noviisi voi arvata indeksiarvojen muutokset yksinkertaisessa forsilmukassa ja määrittää tämän rooliksi *askeltaja*, koska nämä arvot ovat arvioitavissa heti jakson alussa. Toisaalta noviisi voi roolittaa Fibonaccin lukujonon *kokoojaksi*, koska muuttuja saa aina arvonsa edellisten arvojen summasta. Roolit eivät siis ole täysin yksilöitävissä, mutta jotta niitä voitaisiin käyttää opetuksessa tai laajojen ohjelmien analysoinnissa, tulee opettajilla ja tutkijoilla olla käytössään kaikkien hyväksymä kokonaisuus, johon tukeutua (Sajaniemi *et al.*, 2006).

Sajaniemen (2002b) tekemässä tutkimuksessa todettiin, että valtaosa ohjelmistoasiantuntijoiden noviiseille kirjoittamien proseduraalisten ohjelmien muuttujista voidaan luonnehtia yhdeksällä eri roolilla muuttujien roolitussääntöjen version 1.0 mukaisesti (taulukko 2). Byckling *et al.* (2005) jatkoivat roolien tutkimista ja roolittivat kaikki muuttujat kahdesta Java-ohjelmointia opettavasta kirjasta (Peltomäki & Malmirae, 1999; Wikla, 2003) siten, että kaksi tutkijaa analysoivat erikseen kaikki kirjoissa esiintyvät muuttujat ja kävivät lopuksi roolituslistat läpi keskustellen niissä olevista eroista kunnes pääsivät yhteisymmärrykseen kunkin muuttujan roolista. *Kiintoarvojen* korkea määrä olio-ohjelmissa johtui parametreistä, jotka välittävät tietoa oliolta toiselle. Kaikista olio-ohjelmien tutkituista muuttujista tällaisia *kiintoarvoja* oli 43 %. Kun näitä tutkimustuloksia verrattiin aikaisemmin tehtyihin kolmesta Pascal-ohjelmointia opettavasta kirjasta (Jones, 1982; Sajaniemi & Karjalainen, 1985; Foley, 1991) saatuihin roolitustuloksiin (Sajaniemi, 2002b), huomattiin, että vähäisempi tarve *askeltajille* esimerkiksi funktionaalisiin ohjelmiin verrattuna johtui olio-ohjelmoinnin tavasta enkapsuloida silmukoiden ohjausmuuttujat iteraattoreihin. Lisäksi tutkimuksessa todettiin, että olio-ohjelmien tutkimiseen tarvitaan joitakin uusia rooleja (Byckling *et al.*, 2005).

Eri ohjelmointiparadigmoihin perustuvissa oppikirjoissa huomattiin olevan selkeitä eroja siinä, mitä toimintoja kussakin kirjassa esiintyvät ohjelmat suorittivat (Sajaniemi *et al.*, 2006). Tutkimuksessa käytettyjen olio-ohjelmointia opettavien kirjojen taipumuksena oli esittää staattisia tietojen välisiä suhteita, mikä myös osaltaan joh-

ti *kiintoarvojen* suureen esiintymistiheyteen. *Kokoojien* ja *sopivimman säilyttäjien* suhteellisen pieni esiintymistiheys johtui algoritmien mielekkäitä tuloksia laskevien olio-ohjelmien vähäisestä määrästä. Olio-ohjelmointikielien graafisten ominaisuuksien esittely vaati puolestaan hyvin vähän syötteitä, ja näin myös syötteiden käsittelyyn käytettyjen *tuoreimman säilyttäjien* esiintymistiheys laski. *Yksisuuntaisten lippujen* suuri määrä olio-ohjelmissa johtui animaation käsittelyssä tarvittavien lopetusmerkkien (stop flag) käytöstä ja käyttäjien yksinkertaisten toimien seuraamiseen käytetyistä muuttujista. Lisäksi olio-ohjelmien analysoinnissa tarvittavien uusien roolien (Byckling *et al.*, 2005) todettiin olevan *kulkija* ja *säiliö* (Sajaniemi *et al.*, 2006). Tutkimuksessa todettiinkin tämän uuden 11 roolin roolijaon muuttujien roolitussääntöjen version 2.0 mukaisesti (taulukko 5) kattavan käytännössä kaikki muuttujat yksinkertaisessa ohjelmoinnissa, ja että uutta roolijakoa voidaan käyttää myös kaikkien kolmen ohjelmointiparadigman muuttujien roolien tutkimisessa ja opettamisessa.

Muuttujien rooleja olio-ohjelmoinnissa aikaisemmin tutkittaessa on siis arvioitu, että tutkimuksessa käyttämäni version 1.2 mukainen roolijako alarooleineen (taulukko 4) olisi riittävä, mutta oikean roolin löytäminen muuttujille voi tietyissä tapauksissa olla vaikeaa, jos kyseessä on esimerkiksi *muuntajatyyppinen* muuttuja (Sajaniemi *et al.*, 2006). *Muuntaja*-roolin tarkempi tutkiminen osoittikin sen olevan useimmissa tapauksissa tarpeeton, sillä se voidaan roolittaa joko *väliaikaissäilöksi* tai, jos muuttujan elinikä on pidempi, *muuntaja* voidaan roolittaa *tuoreimman säilyttäjäksi*. Olio-ohjelmissa esiintyvien muuttujien roolien analysointiin soveltuvatkin siis vielä paremmin uusimmat version 2.0 mukaiset roolitussäännöt (taulukko 5). Yhdyn myös tutkimuksen tekijöiden mielipiteeseen, että ohjelmakoodin kirjoittajan tekemä muuttujien roolien esittäminen kommenttien muodossa helpottaisi ohjelmien ymmärtämistä ja siten myös muuttujien roolien analysointia. Roolien analysointiin liittyvien aikaisempien tutkimusten tarkemmat tulokset on esitelty tutkielmani tulososan kohdassa 4.2, ja niitä on myös verrattu tutkimuksestani saatuihin tuloksiin.

Roolien analysointiin eri ohjelmointikielissä ja paradigmoissa liittyvän tutkimuksen lisäksi muuttujien roolien on todettu olevan intuitiivisia ja opettajien helposti omaksuttavissa (Ben-Ari & Sajaniemi, 2004). Oppilaiden on todettu pystyvän ymmärtämään roolit, kun tätä käsitettä on käytetty opetuksen apuvälineenä (Kuittinen & Sajaniemi, 2004), ja roolien on todettu olevan osa ohjelmistoasiantuntijoiden hiljaisesta ohjelmointitietämyksestä (Sajaniemi & Navarro Prieto, 2005).

3 Aineisto ja tutkimusmenetelmät

Aineisto ja tutkimusmenetelmät -luvussa käsitellään ensimmäisessä kohdassa 3.1 tutkimuksen taustoja ja päämääriä, 3.2 esittelee tutkimusaineiston valintaa, kohdassa 3.3 esitetään aineiston läpikäynti ja muuttujien roolitustavat erityisesti olioiden roolituksen osalta ja lopuksi esitellään, mitä tilastollisia analyysejä tutkielman teossa on käytetty.

3.1 Tutkimuksen taustat ja päämäärä

Tutkimuksen tavoitteena on ohjelmistoasiantuntijoiden Javaan perustuvien olio-ohjelmien muuttujien roolien analysointi. Päämääränä on selvittää, soveltuuko nykyinen rooliluokitus myös ohjelmistoasiantuntijoiden kirjoittamiin ohjelmiin ja erityisesti olioparadigmaan perustuvien ohjelmien roolitukseen. Tutkimuksessa selvitetään myös, ovatko nykyiset rooliluokitukset riittäviä, voiko joitakin rooleja yhdistää uudeksi yläluokaksi, johon tietyt vanhan roolijaon mukaiset roolit voisi yhdistää, vai pitäisikö jokin rooli jopa jakaa useampaan alaluokkaan. Lisäksi ohjelmien parametreistä määritetään, ovatko ne kaikki alkuoletuksen mukaisesti *kiintoarvoja* (fix pa), vai löytyykö ohjelmista myös muuntyyppisiä parametrejä.

Ohjelmistoasiantuntijoiden kirjoittamista ohjelmista saatuja tuloksia verrataan myös jo aikaisemmin analysoituihin ohjelmistoasiantuntijoiden tekemiin mutta aloitteleville ohjelmoijille suunnattuihin Java-, Pascal- ja ML-ohjelmiin.

3.2 Aineiston valinta

Aineiston valinnan ensimmäinen kriteeri oli, että ohjelmien tuli olla olioparadigmaan perustuvia Java-ohjelmia. Lähdekoodin tuli olla vapaasti saatavilla ja käytettävissä. Tästä johtuen ohjelmat päätettiin valita Sourceforge.net-sivuston avointen lähdekoodien joukosta. Lisäksi ohjelmissa täytyi olla tuhansia rivejä koodia ja satoja muuttujia.

Ensimmäinen analysoitu ohjelma MegaMek-v0.30.7 on Javalla toteutettu vuoropohjainen asiakas-palvelin-malliin pohjautuva strategiapeli, joka perustuu BattleTech-lautapeliin ja sen sääntöihin (SourceForge, 2008). Ohjelmaa on kehitetty vuodesta 2000 alkaen, ja nykyisin projektissa on mukana yli 30 kehittäjää. Ohjelmakokonai-

suudesta valittiin tutkittavaksi keskeinen server.java-tiedosto, joka sisältää lähes kaiken pelimekaniikan ja asiakkaiden välisen liikenteen käsittelyn. Ohjelman tämä osa sisältää 16900 riviä koodia, 1646 muuttujaa sekä 480 parametria.

Toinen analysoitu ohjelma Hsqldb_1_8_0_8 on muun muassa OpenOffice.org-toimisto-ohjelmistossa käytetty Javalla toteutettu relaatiotietokannan tietokantamoottori (relational database engine) ja pohjautuu jo lopetettuun HypersonicSQL-ohjelmaan (SourceForge, 2008). Ohjelmaa on kehitetty vuodesta 2001 alkaen, ja nykyisin projektissa on mukana yli 30 kehittäjää. Ohjelmakokonaisuudesta valittiin tutkittavaksi SqlFile.java-tiedosto, joka käsittelee käyttäjän tietokantaan tekemien kyselyjen prosessoinnin kuten vajavaisten SQL-käskyjen hylkäämisen. Ohjelman tämä osa sisältää 4834 riviä koodia ja 320 muuttujaa, parametrien määrää ei tutkittu.

Kolmas analysoitu ohjelma Azureus on Javalla toteutettu BitTorrentiin pohjautuva asiakasohjelma (SourceForge, 2008). Ohjelmaa on kehitetty vuodesta 2003 alkaen, ja nykyisin projektissa on mukana noin 30 kehittäjää. Ohjelmakokonaisuudesta valittiin tutkittavaksi ListView.java, joka käsittelee kuvien ja muun informaation näyttämistä ja käsittelyä listamuodossa näyttöruudulla. Ohjelman tämä osa sisältää 3663 riviä koodia ja 458 muuttujaa. Toiseksi käsiteltäväksi osaksi otettiin TableViewSWTImpl.java, joka käsittelee kuvien ja muun informaation näyttämistä ja käsittelyä taulukkomuodossa näyttöruudulla. Ohjelman tämä osa sisältää 3631 riviä koodia ja 374 muuttujaa, parametrien määrää ei tutkittu.

Yllämainittujen lisäksi vertailuaineistona on yhdeksän aikaisemmin analysoitua ohjelmistoasiantuntijoiden tekemää mutta aloitteleville ohjelmoijille (novice level) suunnattua ohjelmaa, joiden lähdeaineisto on saatu yhdeksän eri oppikirjan esimerkkiohjelmista (Sajaniemi, 2002b; Byckling *et al.*, 2005; Kulikova, 2005). Näistä yhdeksästä oppikirjasta kaksi on Java-kielen oppaita, joista ensimmäisessä analysoitavia muuttujia oli 659 (Peltomäki & Malmirae, 1999) ja toisessa 302 (Wikla, 2003). Kolmesta Pascal-kielen oppaasta ensimmäisessä analysoitavia muuttujia oli 136 (Sajaniemi & Karjalainen, 1985), toisessa 136 (Foley, 1991) ja kolmannessa 154 (Jones, 1982). Neljästä ML-kielen oppaasta ensimmäisessä analysoitavia muuttujia oli 361 (Ullman, 1998), toisessa 1334 (Paulson, 1996), kolmannessa 806 (Hansen & Rischel, 1999) ja neljännessä 964 (Michaelson, 1995).

3.3 Aineiston läpikäynti ja muuttujien roolitus

Ohjelmat käytiin tutkielman tekijän toimesta lävitse kahdesti siten, että ensimmäisellä kerralla kaikki muuttujat ja parametrit kirjattiin ylös mahdollisimman tarkasti ja toisella kerralla syntynyttä listaa täydennettiin vielä puuttuvilla esiintymillä. Kunkin ohjelman muuttujakokonaisuudesta valittiin satunnaisesti (Selection of Non-Repeated Numbers from Sample) 150 muuttujaa roolitettavaksi (Byers, 1991). Lisäksi ensimmäisestä analysoitavasta ohjelmasta (MegaMek-v0.30.7 server.java -tiedosto) valittiin kaikkien 480 parametrin joukosta satunnaisesti 250 parametria roolitettavaksi. Muuttujat roolitettiin käyttäen taulukosta 4 (Sajaniemi, 2004) löytyvää version 1.2 mukaista roolijakoa.

Edelläolevan lisäksi uusien roolien toimivuuden todentamiseksi ohjelmistoasiantuntijoiden kirjoittamissa Java-ohjelmissa sekä Java- että Pascal-oppikirjoissa esiintyvät muuttujien roolit muunnettiin taulukossa 5 esiintyvien uusimpien roolitussääntöjen version 2.0 mukaisiksi siten, että *muuntaja*-alaroolit (*trn fc, fs, fe*) muunnettiin *kiintoarvoiksi* (*fix*), *muuntaja*-alaroolit (*trn rc, rs, re*) muunnettiin *tuoreimman säilyttäjiksi* (*mrh*) sekä *muut*-alaryhmä (*oth ar*) muunnettiin *säiliöiksi* (*cnt*).

3.3.1 Olioiden roolituksen erikoissäännöt

Niin sanottujen perinteisten muuttujien roolitus ei eroa Javassa ei-oliopohjaisista proseduraalisista paradigmoista kuten C-kielestä. Suuri osa Javan muuttujista on kuitenkin olioita, joiden roolittamiseen käytetään tutkimuksen aikana luotuja erikoissääntöjä.

Jos olioviite tarkoittaa elämänsä aikana eri olioita, se roolitetaan osoittimena. Esimerkiksi kuvan 4 ohjelmassa muuttuja *y* saa arvokseen listan kussakin kohdassa esiintyvän olion, kunnes se lopulta saavuttaa listan lopun ja saa arvokseen *null*-elementin. Koska *y* tarkoittaa elinaikanaan eri olioita, se roolitetaan osoittimena ja on tässä kyseisessä esimerkissä rooliltaan *kulkija*.

Kun olioviite tarkoittaa koko elinaikansa samaa oliota ja on *kiintoarvo*, käytetään seuraavia erikoissääntöjä.

Jos viitattu olio on ”tavallinen” olio, roolitetaan viitatun olion attribuutit erikseen. Esimerkiksi kuvan 5 ohjelmassa olion (*evil*) attribuutti ”evil.name” saa arvokseen kiin-

```
public int length() {
  List y = this;
  int len = 0;
  while (y != null) {
    len++;
    y = y.next;
  }
return len;
}
```

Kuva 4: Esimerkki muuttujasta (y), joka on *kulkija*.

toarvon *n* ja on siten rooliltaan *kiintoarvo*. Attribuutin ”evil.age” arvo ei pysy vakiona vaan kasvaa aina, kun metodia *birthday* kutsutaan, ja siten attribuutti ”evil.age” on rooliltaan *kokooja*.

Jos viitattu olio kätkee yhden käsitteellisen attribuutin, olioviite roolitetaan kyseisen käsitteellisen attribuutin roolin mukaan. Esimerkiksi kuvan 6 ohjelmassa String-tyyppinen olio *jono* saa arvokseen merkkijonon ”tekstiä”, ja tätä merkkijonoa kasvatetaan while-silmukassa lisäämällä joka suorituskerralla sen perään yksittäinen merkki ”ä”. Koska olioon *jono* kerätään merkkejä, se roolitetaan *kokoojaksi*.

Jos viitattu olio on tietorakenne, jonka kaikkien alkioiden rooli on sama, olioviitteen rooliksi otetaan tämä rooli. Esimerkiksi kuvan 7 ohjelmassa int[]-tyyppinen olio *kk_myynti* saa arvokseen 12 kokonaislukua sisältävän taulun, jonka alkioihin kerätään while-silmukassa aina kunkin kuun myyntitapahtuman kokonaismuutos. Koska tietorakenteen kaikki 12 alkiota ovat rooliltaan *kokoojia*, myös *kk_myynti*-olion rooli on *kokooja*.

Jos viitattu olio on muu tietorakenne, olioviitteen rooliksi otetaan kyseisen tietorakenteen rooli. Esimerkiksi kuvan 8 ohjelmassa DefaultMutableTreeNode-tyyppinen olio *node* saa arvokseen olion, joka edustaa kulloinkin valittuna olevaa puun solmua. Jos *node*-olion todetaan pitävän muistissa aina puun senhetkistä solmua, kun puuta käydään läpi, se roolitetaan *kulkijaksi*, jos taas olion katsotaan pitävän muistissa aina viimeisintä arvoa, kun puuta käydään läpi, se roolitetaan *tuoreimman säilyttäjäksi*.

```
public class Ferret {
  String name;
  int age;

  public Ferret (String n) {
    name = n;
    age = 0;
  }

  public void birthday () {
    age++;
  }
}

Ferret evil;
```

Kuva 5: Esimerkki oliosta (*ferret*), jonka attribuuteista ”evil.name” on rooliltaan *kiinto-arvo* ja ”evil.age” *kokooja*.

```
String jono = new String("tekstiä");

while (...) {
  jono = jono + "ä";
}
```

Kuva 6: Esimerkki oliosta (*jono*), joka on rooliltaan *kokooja*.

```
int[] kk_myynti = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

while (...) {
  kk_myynti[kuu] += myyntitapahtuma;
}
```

Kuva 7: Esimerkki oliosta (*kk_myynti*), joka on rooliltaan *kokooja*.

```
class MyTreeModelListener implements TreeModelListener {
  public void treeNodesChanged(TreeModelEvent e) {
    DefaultMutableTreeNode node;
    node = (DefaultMutableTreeNode)
            (e.getTreePath().getLastPathComponent());
    try {
      int index = e.getChildIndices()[0];
      node = (DefaultMutableTreeNode)
              (node.getChildAt(index));
    } catch (NullPointerException exc) {}
  }
}
```

Kuva 8: Esimerkki oliosta (*node*), joka on rooliltaan *kulkija* tai *tuoreimman säilyttäjä*.

3.4 Tilastollinen analyysi

Tutkimuksesta saatu aineisto käsiteltiin awk-ohjelmointikielellä tehdyllä BASICS-ohjelmalla (Sajaniemi, 2003), joka laski jokaisesta roolista ja alaroolista niiden lukumäärän, rooleista prosentuaalisen osuuden kaikista rooleista ja alarooleista prosentuaalisen osuuden kaikista alarooleista sekä prosentuaalisen osuuden roolin sisällä. Lopuksi roolien jakaumaa eri ohjelmissa verrattiin SPSS-tilasto-ohjelman (SPSS Inc., Chicago, Illinois, Yhdysvallat) χ^2-testillä.

4 Tulokset

Seuraavissa kahdessa luvussa käsitellään tutkimuksesta saatuja tuloksia lukumäärien, prosentuaalisten osuuksien ja jakaumien osalta. Kohdassa 4.1 käsitellään ohjelmistoasiantuntijoiden kirjoittamista kolmesta Java-ohjelmasta saatuja lukumääräisiä ja prosentuaalisia tuloksia, kun taas kohdassa 4.2 käsitellään näissä ja kolmessa aikaisemmassa tutkimuksessa (Sajaniemi, 2002b; Byckling *et al.*, 2005; Kulikova, 2005) esiintyneiden muuttujien roolien jakaumien eroja.

4.1 Muuttujien roolit ohjelmistoasiantuntijoiden ohjelmissa

Megamek-ohjelman 480 parametrin joukosta tutkittujen 250 parametrin otoksen kattavuus oli 52,1 %, ja kaikkien otokseen kuuluvien parametrien rooli oli *kiintoarvo* (fix pa). Parametrit käsiteltiin erillisenä kokonaisuutena eivätkä ne sisälly jatkossa käsiteltäviin tulosjoukkoihin.

Taulukossa 6 ovat Megamek-ohjelman valituissa osissa esiintyvien muuttujien roolien ja alaroolien lukumäärät, prosentuaaliset osuudet kaikista rooleista sekä alaroolien kohdalla alaroolin osuus kyseisestä roolista muuttujien roolitussääntöjen version 1.2 mukaisesti (taulukko 4). Tutkittujen 150 muuttujan otoksen kattavuus 1646 muuttujan kokonaismäärästä oli 9,1 % muuttujien esiintymistiheyden ollessa 9,8 muuttujaa jokaista ohjelmakoodin 100 riviä kohden, kun ohjelmassa oli 16900 riviä. Tutkituista muuttujista suurin osa oli tyypiltään *kiintoarvoja* (fix) näiden osuuden ollessa kokonaismäärästä 38 kappaletta (25,3 %). Toiseksi suurin ryhmä oli 32 esiintymällään (21,3 %) *tuoreimman säilyttäjät* (mrh) ja kolmantena lähes yhtä suurella osuudella *muuntaja*-rooli 31 esiintymällään (20,7 %). Muiden roolien osuus jäi 32,7 prosenttiin. *Askeltajia* (stp) oli 18 (12,0 %), *yksisuuntaisia lippuja* (one) 14 (9,3 %), *tilapäissäilöjä* (tmp) 11 (7,3 %) ja *kokoojia* (gat) 6 (4,0 %). Ilman esiintymiä jäivät *järjestelijät* (org), *seuraajat* (fol) sekä mihinkään edellämainittuihin ryhmiin kuulumattomat *muut* (oth).

Taulukko 6: Muuttujien roolit Megamek-ohjelmassa.

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fix		38	25.33	
fix	in	0	0.00	0.00
fix	ic	1	0.67	2.63
fix	co	6	4.00	15.79
fix	va	8	5.33	21.05
fix	pa	3	2.00	7.89
fix	dv	7	4.67	18.42
fix	ot	13	8.67	34.21
org		0	0.00	
org	so	0	0.00	0.00
org	re	0	0.00	0.00
org	ra	0	0.00	0.00
org	ot	0	0.00	0.00
stp		18	12.00	
stp	cn	1	0.67	5.56
stp	i1	9	6.00	50.00
stp	in	0	0.00	0.00
stp	nc	0	0.00	0.00
stp	d1	0	0.00	0.00
stp	dn	0	0.00	0.00
stp	a2	0	0.00	0.00
stp	an	0	0.00	0.00
stp	ll	8	5.33	44.44
stp	ot	0	0.00	0.00
mrh		32	21.33	
mrh	in	0	0.00	0.00
mrh	ic	1	0.67	3.12
mrh	ae	28	18.67	87.50
mrh	dv	3	2.00	9.38
mrh	ot	0	0.00	0.00

Taulukko 6: Muuttujien roolit Megamek-ohjelmassa (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
gat		6	4.00	
gat	su	1	0.67	16.67
gat	de	2	1.33	33.33
gat	mu	0	0.00	0.00
gat	di	0	0.00	0.00
gat	ot	3	2.00	50.00
mwh		0	0.00	
mwh	ma	0	0.00	0.00
mwh	mi	0	0.00	0.00
mwh	cl	0	0.00	0.00
mwh	fa	0	0.00	0.00
mwh	fi	0	0.00	0.00
mwh	la	0	0.00	0.00
mwh	ot	0	0.00	0.00
one		14	9.33	
one	r1	6	4.00	42.86
one	rn	0	0.00	0.00
one	l1	1	0.67	7.14
one	ln	0	0.00	0.00
one	ot	7	4.67	50.00
trn		31	20.67	
trn	fc	6	4.00	19.35
trn	rc	0	0.00	0.00
trn	fs	0	0.00	0.00
trn	rs	0	0.00	0.00
trn	fe	24	16.00	77.42
trn	re	0	0.00	0.00
trn	ot	1	0.67	3.23
fol		0	0.00	
fol	st	0	0.00	0.00
fol	ns	0	0.00	0.00

Taulukko 6: Muuttujien roolit Megamek-ohjelmassa (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fol	ot	0	0.00	0.00
tmp		11	7.33	
tmp	sw	0	0.00	0.00
tmp	ot	11	7.33	100.00
oth		0	0.00	
oth	sc	0	0.00	0.00
oth	ar	0	0.00	0.00
oth	ot	0	0.00	0.00
yht		150	100	

Roolitussääntöjen version 1.1 (taulukko 3) mukaisesti ilman alaroolijaottelua ja roolien yleisyysjärjestyksessä esitetyt tulokset ohjelmatyypeittäin löytyvät taulukosta 7. Vastaavat tulokset ohjelmittain löytyvät taulukosta 8.

Taulukko 7: Roolit ohjelmatyypeittäin roolitussääntöjen version 1.1 mukaan. (1: ei tutkittu.)

Rooli	*Java/Ekspertit*	*Java/Noviisit*	*Pascal/Noviisit*	*ML/Noviisit*
fix	25.3–32.7 %	43.7–67.4 %	18.2–39.7 %	28.7–42.7 %
stp	12.0–18.0 %	14.0–20.9 %	21.3–37.2 %	9.6–16.5 %
mrh	14.7–22.7 %	3.3–8.6 %	14.0–16.4 %	15.5–22.0 %
trn	9.3–20.7 %	8.2–14.9 %	10.4–15.6 %	13.1–18.5 %
gat	1.3–4.0 %	1.4–1.7 %	5.1–6.7 %	10.3-15.4 %
one	5.3–9.3 %	3.0–3.3 %	0.0–1.9 %	0.0–1.1 %
tmp	0.7–7.3 %	0.2–1.0 %	0.7–4.5 %	–[1]
org	0.0–1.3 %	0.0–1.0 %	0.7–4.5 %	0.5–1.9 %
mwh	0.0–2.0 %	0.0-0.9 %	1.1–2.9 %	0.0–0.8 %
fol	0.0–0.0 %	0.7–1.2 %	0.0–3.3 %	0.0–1.0 %
oth	0.0–10.0 %	0.5–4.3 %	0.0–1.9 %	0.0–0.1 %

Taulukko 8: Roolien prosenttiosuudet ohjelmittain roolitussääntöjen version 1.1 mukaan.

Rooli	Megamek	Hsqldb	Azureus	Sajaniemi & Karjalainen, 1985	Foley, 1991	Jones, 1982	Peltomäki & Malmirae, 1999	Wikla, 2003	Ullman, 1998	Paulson, 1996	Hansen, 1999	Michaelson, 1995
fix	25.33 %	32.67 %	28.67 %	39.71 %	25.97 %	18.22 %	67.37 %	43.71 %	28.68 %	31.17 %	42.68 %	35.72 %
stp	12.00 %	15.33 %	18.00 %	21.32 %	25.97 %	37.17 %	13.96 %	20.86 %	16.35 %	12.75 %	9.55 %	16.54 %
mrh	21.33 %	14.67 %	22.67 %	13.97 %	14.29 %	16.36 %	3.34 %	8.61 %	20.11 %	22.00 %	15.51 %	16.63 %
trn	20.67 %	14.00 %	9.33 %	13.24 %	15.58 %	10.41 %	8.19 %	14.90 %	18.50 %	13.11 %	16.25 %	14.17 %
gat	4.00 %	1.33 %	4.00 %	5.15 %	5.84 %	6.69 %	1.37 %	1.66 %	10.46 %	15.37 %	12.16 %	10.33 %
one	9.33 %	8.00 %	5.33 %	1.47 %	0.00 %	1.86 %	3.03 %	3.31 %	0.00 %	0.66 %	1.12 %	0.39 %
tmp	7.33 %	0.67 %	2.67 %	0.74 %	4.55 %	0.74 %	0.15 %	0.99 %	0.00 %	0.00 %	0.00 %	0.00 %
org	0.00 %	1.33 %	0.67 %	0.74 %	4.55 %	2.23 %	0.00 %	0.99 %	1.88 %	0.66 %	0.50 %	0.59 %
mwh	0.00 %	2.00 %	2.67 %	2.94 %	1.95 %	1.12 %	0.91 %	0.00 %	0.80 %	0.44 %	0.00 %	0.49 %
fol	0.00 %	0.00 %	0.00 %	0.00 %	1.30 %	3.35 %	1.21 %	0.66 %	0.00 %	1.02 %	0.50 %	0.00 %
oth	0.00 %	10.00 %	6.00 %	0.74 %	0.00 %	1.86 %	0.46 %	4.30 %	0.00 %	0.00 %	0.12 %	0.12 %

Uusimpien roolitussääntöjen version 2.0 (taulukko 5) mukaisesti ilman alaroolijaottelua ja roolien yleisyysjärjestyksessä esitetyt tulokset ohjelmatyypeittäin löytyvät taulukosta 9. Vastaavat tulokset ohjelmittain löytyvät taulukosta 10.

Taulukko 9: Roolit ohjelmatyypeittäin roolitussääntöjen version 2.0 mukaan ilman ML-kielisiä ohjelmia, joille ei pystytty tekemään roolituksien muunnosta uusimpien roolitussääntöjen mukaisesti.

Rooli	*Java/Ekspertit*	*Java/Noviisit*	*Pascal/Noviisit*
fix	36.0–46.0 %	57.6–75.0 %	23.0–48.5 %
stp	12.0–18.0 %	14.0–20.9 %	21.3–37.2 %
mrh	16.0–24.7 %	3.9–9.6 %	18.4–20.1 %
one	5.3–9.3 %	3.0–3.3 %	0.0–1.9 %
gat	1.3–4.0 %	1.4–1.7 %	5.1–6.7 %
cnt	0.0–8.0 %	0.3–2.6 %	0.0–1.1 %
tmp	0.7–7.3 %	0.2–1.0 %	0.7–4.5 %
mwh	0.0–2.7 %	0.0-0.9 %	1.1–2.9 %
org	0.0–1.3 %	0.0–1.0 %	0.7–4.5 %
fol	0.0–0.0 %	0.7–1.2 %	0.0–3.3 %
oth	0.0–2.0 %	0.2–1.7 %	0.7–2.6 %

Taulukko 10: Roolien prosenttiosuudet ohjelmittain roolitussääntöjen version 2.0 mukaan ilman ML-kielisiä ohjelmia, joille ei pystytty tekemään roolituksien muunnosta uusimpien roolitussääntöjen mukaisesti.

Rooli	Megamek	Hsqldb	Azureus	Sajaniemi & Karjalainen, 1985	Foley, 1991	Jones, 1982	Peltomäki & Malmirae, 1999	Wikla, 2003
fix	46.0 %	45.33 %	36.00 %	48.53 %	35.06 %	23.05 %	74.96 %	57.62 %
stp	12.00 %	15.33 %	18.00 %	21.32 %	25.97 %	37.17 %	13.96 %	20.86 %
mrh	21.33 %	16.00 %	24.67 %	18.38 %	19.48 %	20.07 %	3.95 %	9.60 %
one	9.33 %	8.00 %	5.33 %	1.47 %	0.00 %	1.86 %	3.03 %	3.31 %
gat	4.00 %	1.33 %	4.00 %	5.15 %	5.84 %	6.69 %	1.37 %	1.66 %
cnt	0.00 %	8.00 %	6.00 %	0.00 %	0.00 %	1.12 %	0.30 %	2.65 %
tmp	7.33 %	0.67 %	2.67 %	0.74 %	4.55 %	0.74 %	0.15 %	0.99 %
mwh	0.00 %	2.00 %	2.67 %	2.94 %	1.95 %	1.12 %	0.91 %	0.00 %
org	0.00 %	1.33 %	0.67 %	0.74 %	4.55 %	2.23 %	0.00 %	0.99 %
fol	0.00 %	0.00 %	0.00 %	0.00 %	1.30 %	3.35 %	1.21 %	0.66 %
oth	0.00 %	2.00 %	0.00 %	0.74 %	1.30 %	2.60 %	0.15 %	1.66 %

Megamek-ohjelmassa version 2.0 mukaisista rooleista (kts. taulukko 10) suurin ryhmä oli *kiintoarvot* 69 esiintymällä (46,0 %), eli kiintoarvojen osuus kasvoi 20,7 % roolitussääntöjen version 1.2 mukaisiin roolituksiin verrattuna. *Tuoreimman säilyttäjien* lukumäärä ei muuttunut, ja ne pysyivät toiseksi suurimpana ryhmänä. Myös *askeltajien* lukumäärä säilyi samana, mutta muista roolituksessa tapahtuneista muutoksista johtuen ryhmä nousi kolmanneksi suurimmaksi rooliksi. Muiden uusimpiin roolitussääntöihin kuuluvien roolien osuudet eivät muuttuneet, mutta mukaan tuli *säiliö*-rooli, joka jäi kuitenkin ilman esiintymiä.

Hsqldb-ohjelman roolien ja alaroolien lukumäärät, prosentuaaliset osuudet kaikista rooleista sekä alaroolien osuus roolin sisällä on esitelty taulukossa 11 muuttujien roolitussääntöjen version 1.2 mukaisesti (taulukko 4). Ohjelmasta tutkittujen 150 muuttujan otos oli 320 muuttujan kokonaismäärästä 48,7 % ja muuttujien esiintymistiheys 6,6 muuttujaa jokaista ohjelmakoodin 100 riviä kohden ohjelman koon ollessa 4834 riviä. Kolme suurinta ryhmää olivat *kiintoarvot*, *askeltajat* ja *tuoreimman säilyttäjät* esiintymien lukumäärien ja prosentuaalisten osuuksien ollessa 49 (32,7 %), 23 (15,3%) ja 22 (14,7 %). *Askeltajien* ja *tuoreimman säilyttäjien* kanssa lähes yhtä suuri ryhmä oli *muuntajat* 21 esiintymällään ja 14,0 % osuudellaan. Seitsemän jäljellejääneen roolin osuudet olivat suuruusjärjestyksessä: *askeltajat* 17 (11,3 %), *muut* 15 (10,0 %), *yksisuuntaiset liput* 12 (8,0 %), *sopivimman säilyttäjät* 3 (2,0 %), *kokoojat* ja *järjestelijät* 2 (1,3 %), *tilapäissäilöt* 1 (0,7 %) sekä ilman esiintymiä jääneet *seuraajat*. *Muut*-ryhmän kaikki 12 (oth ar) esiintymää roolitettiin myöhemmin uusimpien 2.0 roolitussääntöjen mukaisesti *säiliöiksi* eli tietorakenteiksi, joihin voidaan lisätä ja joista voidaan poistaa alkioita.

Taulukko 11: Muuttujien roolit Hsqldb-ohjelmassa.

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fix		49	32.67	
fix	in	1	0.67	2.04
fix	ic	0	0.00	0.00
fix	co	13	8.67	26.53
fix	va	21	14.00	42.86
fix	pa	6	4.00	12.24
fix	dv	6	4.00	12.24
fix	ot	2	1.33	4.08

Taulukko 11: Muuttujien roolit Hsqldb-ohjelmassa (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
org		2	1.33	
org	so	1	0.67	50.00
org	re	0	0.00	0.00
org	ra	0	0.00	0.00
org	ot	1	0.67	50.00
stp		23	15.33	
stp	cn	4	2.67	17.39
stp	i1	17	11.33	73.91
stp	in	0	0.00	0.00
stp	nc	0	0.00	0.00
stp	d1	1	0.67	4.35
stp	dn	0	0.00	0.00
stp	a2	0	0.00	0.00
stp	an	0	0.00	0.00
stp	ll	1	0.67	4.35
stp	ot	0	0.00	0.00
mrh		22	14.67	
mrh	in	0	0.00	0.00
mrh	ic	0	0.00	0.00
mrh	ae	16	10.67	72.73
mrh	dv	4	2.67	18.18
mrh	ot	2	1.33	9.09
gat		2	1.33	
gat	su	1	0.67	50.00
gat	de	0	0.00	0.00
gat	mu	0	0.00	0.00
gat	di	0	0.00	0.00
gat	ot	1	0.67	50.00
mwh		3	2.00	
mwh	ma	0	0.00	0.00
mwh	mi	0	0.00	0.00
mwh	cl	0	0.00	0.00

Taulukko 11: Muuttujien roolit Hsqldb-ohjelmassa (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
mwh	fa	0	0.00	0.00
mwh	fi	0	0.00	0.00
mwh	la	3	2.00	100.00
mwh	ot	0	0.00	0.00
one		12	8.00	
one	r1	5	3.33	41.67
one	rn	2	1.33	16.67
one	l1	1	0.67	8.33
one	ln	1	0.67	8.33
one	ot	3	2.00	25.00
trn		21	14.00	
trn	fc	5	3.33	23.81
trn	rc	1	0.67	4.76
trn	fs	1	0.67	4.76
trn	rs	0	0.00	0.00
trn	fe	13	8.67	61.90
trn	re	1	0.67	4.76
trn	ot	0	0.00	0.00
fol		0	0.00	
fol	st	0	0.00	0.00
fol	ns	0	0.00	0.00
fol	ot	0	0.00	0.00
tmp		1	0.67	
tmp	sw	0	0.00	0.00
tmp	ot	1	0.67	100.00
oth		15	10.00	
oth	sc	0	0.00	0.00
oth	ar	12	8.00	80.00
oth	ot	3	2.00	20.00
yht		150	100	

Roolitussääntöjen version 2.0 (taulukko 5) mukaan esitetyissä tuloksissa (kts. taulukko 10) Hsqldb-ohjelman *kiintoarvojen* lukumäärä kasvoi roolitussääntöjen version 1.2 *muuntaja*-rooleista muunnetuilla 19 esiintymällä (12,6 %) nousten näin 68 esiintymään (45,3 %). *Tuoreimman säilyttäjät* nousivat roolitussääntöjen version 1.2 *muuntaja*-rooleista saaduilla kahdella uudella esiintymällä toiseksi suurimmaksi ryhmäksi 24 esiintymällään (16,0 %). *Askeltajien* esiintyvyys ei muuttunut, ja ne olivat kolmanneksi suurin rooli. Muiden uusimpiin roolitussääntöihin kuuluvien roolien osuudet eivät muuttuneet, mutta mukaan tuli roolitussääntöjen version 1.2 *muut*-ryhmästä (oth ar) otetuilla muuttujilla *säiliö*-rooli 12 esiintymällä (8,0 %), joka oli *yksisuuntaisten lippujen* kanssa neljänneksi yleisin. *Muut*-ryhmän koko pieneni kolmeen esiintymään (2,0 %).

Taulukosta 12 käyvät ilmi muuttujien roolitussääntöjen version 1.2 mukaisesti (taulukko 4) Azureus-ohjelman roolien ja alaroolien prosentuaaliset osuudet kaikista rooleista, niiden lukumäärät sekä alaroolien tapauksessa myös prosenttiosuudet roolin sisällä. Tutkittu 150 otos oli 18,0 % koko ohjelmasta valittujen roolitettavien osien 832 muuttujasta, ja muuttujien esiintymistiheys oli 11,4 muuttujaa ohjelman 100 riviä kohden, kun tutkittujen ohjelmaosien yhteiskoko oli 7294 riviä. Ohjelmassa erottuivat selkeästi kolme suurinta ryhmää, joista suurin oli *kiintoarvot* 43 esiintymällään ja 28,7 % osuudellaan, toiseksi suurin ryhmä *tuoreimman säilyttäjät* 34 esiintymällään ja 22,7 % osuudellaan ja pienin kolmesta suurimmasta ryhmästä *askeltajat* 27 esiintymällään prosentuaalisen osuuden ollessa 18,0. *Muuntajat* oli neljänneksi suurin ryhmä 14 esiintymällä ja 9,3 % osuudella, ja seitsemän jäljellejääneen roolin osuudet olivat suuruusjärjestyksessä: *muut* 9 (6,0 %), *yksisuuntaiset liput* 8 (5,3 %), *kokoojat* 6 (4,00 %), *tilapäissäilöt* ja *sopivimman säilyttäjät* neljällä esiintymällä (2,7 %), *järjestelijät* 1 (0,7 %) sekä ilman esiintymiä jäänyt *seuraajat*-rooli.

Taulukko 12: Muuttujien roolit Azureus-ohjelmassa.

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fix		43	28.67	
fix	in	0	0.00	0.00
fix	ic	0	0.00	0.00
fix	co	13	8.67	30.23
fix	va	8	5.33	18.60

Taulukko 12: Muuttujien roolit Azureus-ohjelmassa (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fix	pa	11	7.33	25.58
fix	dv	9	6.00	20.93
fix	ot	2	1.33	4.65
org		1	0.67	
org	so	1	0.67	100.00
org	re	0	0.00	0.00
org	ra	0	0.00	0.00
org	ot	0	0.00	0.00
stp		27	18.00	
stp	cn	1	0.67	3.70
stp	i1	21	14.00	77.78
stp	in	0	0.00	0.00
stp	nc	0	0.00	0.00
stp	d1	1	0.67	3.70
stp	dn	0	0.00	0.00
stp	a2	0	0.00	0.00
stp	an	0	0.00	0.00
stp	ll	4	2.67	14.81
stp	ot	0	0.00	0.00
mrh		34	22.67	
mrh	in	0	0.00	0.00
mrh	ic	1	0.67	2.94
mrh	ae	32	21.33	94.12
mrh	dv	1	0.67	2.94
mrh	ot	0	0.00	0.00
gat		6	4.00	
gat	su	4	2.67	66.67
gat	de	0	0.00	0.00
gat	mu	0	0.00	0.00
gat	di	0	0.00	0.00
gat	ot	2	1.33	33.33

Taulukko 12: Muuttujien roolit Azureus-ohjelmassa (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
mwh		4	2.67	
mwh	ma	0	0.00	0.00
mwh	mi	0	0.00	0.00
mwh	cl	0	0.00	0.00
mwh	fa	0	0.00	0.00
mwh	fi	0	0.00	0.00
mwh	la	2	1.33	50.00
mwh	ot	2	1.33	50.00
one		8	5.33	
one	r1	2	1.33	25.00
one	rn	0	0.00	0.00
one	l1	1	0.67	12.50
one	ln	0	0.00	0.00
one	ot	5	3.33	62.50
trn		14	9.33	
trn	fc	8	5.33	57.14
trn	rc	0	0.00	0.00
trn	fs	0	0.00	0.00
trn	rs	0	0.00	0.00
trn	fe	3	2.00	21.43
trn	re	3	2.00	21.43
trn	ot	0	0.00	0.00
fol		0	0.00	
fol	st	0	0.00	0.00
fol	ns	0	0.00	0.00
fol	ot	0	0.00	0.00
tmp		4	2.67	
tmp	sw	0	0.00	0.00
tmp	ot	4	2.67	100.00
oth		9	6.00	
oth	sc	0	0.00	0.00

Taulukko 12: Muuttujien roolit Azureus-ohjelmassa (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
oth	ar	9	6.00	100.00
oth	ot	0	0.00	0.00
yht		150	100	

Azureus-ohjelman osalta roolitussääntöjen version 2.0 (taulukko 5) mukaan esitetyissä tuloksissa (kts. taulukko 10) *kiintoarvojen* lukumäärä kasvoi 11 esiintymällä (7,3 %) nousten näin 54 esiintymään (36,0 %). *Tuoreimman säilyttäjien* lukumäärä nousi kolmella esiintymällä (2,0 %) 37 esiintymään (24,7 %). Muiden uusimpiin roolitussääntöihin kuuluvien roolien osuudet eivät muuttuneet, mutta mukaan tuli *muut*-ryhmästä (oth ar) otetuilla muuttujilla *säiliö*-rooli 9 esiintymällä (6,0 %).

4.2 Muuttujien roolien jakaumien erot ohjelmatyypeittäin

Tutkimuksessa kerättyä aineistoa verrattiin lähdemateriaaliin, jota on käytetty osittain kolmessa aikaisemmassa tutkimuksessa (Sajaniemi, 2002b; Byckling *et al.*, 2005; Kulikova, 2005). Vertailun tulokset löytyvät Pascal-ohjelmien osalta taulukoista 13, 14 ja 15, Java-ohjelmien osalta taulukoista 16 ja 17 ja ML-ohjelmien osalta taulukosta 18. Tutkielmassani tehdyn roolimuunnoksen tulokset (ML-kirjoissa esiintyviä roolеja lukuunottamatta) roolitussääntöjen versiosta 1.1 (taulukko 3) roolitussääntöihin 2.0 (taulukko 5) löytyvät taulukosta 10. Kaikki jatkossa esiteltävät p:n arvot ovat χ^2-testin tuloksia.

Taulukko 13: Muuttujien roolit Sajaniemen & Karjalaisen (1985) kirjassa esiintyvissä ohjelmissa (Sajaniemi, 2002b).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fix		54	39.71	
fix	in	45	33.09	83.33
fix	ic	1	0.74	1.85
fix	co	5	3.68	9.26
fix	va	1	0.74	1.85

Taulukko 13: Muuttujien roolit Sajaniemen & Karjalaisen (1985) kirjassa esiintyvissä ohjelmissa (Sajaniemi, 2002b) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fix	pa	0	0.00	0.00
fix	ot	2	1.47	3.70
org		1	0.74	
org	so	1	0.74	100.00
org	re	0	0.00	0.00
org	ra	0	0.00	0.00
org	ot	0	0.00	0.00
stp		29	21.32	
stp	cn	6	4.41	20.69
stp	i1	20	14.71	68.97
stp	in	0	0.00	0.00
stp	nc	0	0.00	0.00
stp	d1	1	0.74	3.45
stp	dn	0	0.00	0.00
stp	a2	2	1.47	6.90
stp	an	0	0.00	0.00
stp	ll	0	0.00	0.00
stp	ot	0	0.00	0.00
mrh		19	13.97	
mrh	in	17	12.50	89.47
mrh	ic	1	0.74	5.26
mrh	ae	1	0.74	5.26
mrh	dv	0	0.00	0.00
mrh	ot	0	0.00	0.00
gat		7	5.15	
gat	su	4	2.94	57.14
gat	de	1	0.74	14.29
gat	mu	2	1.47	28.57
gat	di	0	0.00	0.00

Taulukko 13: Muuttujien roolit Sajaniemen & Karjalaisen (1985) kirjassa esiintyvissä ohjelmissa (Sajaniemi, 2002b) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
gat	ot	0	0.00	0.00
mwh		4	2.94	
mwh	ma	3	2.21	75.00
mwh	mi	1	0.74	25.00
mwh	cl	0	0.00	0.00
mwh	fa	0	0.00	0.00
mwh	fi	0	0.00	0.00
mwh	la	0	0.00	0.00
mwh	ot	0	0.00	0.00
one		2	1.47	
one	r1	2	1.47	100.00
one	rn	0	0.00	0.00
one	l1	0	0.00	0.00
one	ln	0	0.00	0.00
one	ot	0	0.00	0.00
trn		18	13.24	
trn	fc	1	0.74	5.56
trn	rc	2	1.47	11.11
trn	fs	3	2.21	16.67
trn	rs	0	0.00	0.00
trn	fe	8	5.88	44.44
trn	re	4	2.94	22.22
trn	ot	0	0.00	0.00
fol		0	0.00	
fol	st	0	0.00	0.00
fol	ns	0	0.00	0.00
fol	ot	0	0.00	0.00
tmp		1	0.74	
tmp	sw	1	0.74	100.00

Taulukko 13: Muuttujien roolit Sajaniemen & Karjalaisen (1985) kirjassa esiintyvissä ohjelmissa (Sajaniemi, 2002b) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
tmp	ot	0	0.00	0.00
oth		1	0.74	
oth	sc	1	0.74	100.00
oth	ar	0	0.00	0.00
oth	ot	0	0.00	0.00
yht		136	100	

Taulukko 14: Muuttujien roolit Foleyn (1991) kirjassa esiintyvissä ohjelmissa (Sajaniemi, 2002b).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fix		40	25.97	
fix	in	19	12.34	47.50
fix	ic	0	0.00	0.00
fix	co	5	3.25	12.50
fix	va	0	0.00	0.00
fix	pa	16	10.39	40.00
fix	dv	0	0.00	0.00
fix	ot	0	0.00	0.00
org		7	4.55	
org	so	7	4.55	100.00
org	re	0	0.00	0.00
org	ra	0	0.00	0.00
org	ot	0	0.00	0.00
stp		40	25.97	
stp	cn	9	5.84	22.50
stp	i1	29	18.83	72.50
stp	in	1	0.65	2.50
stp	nc	0	0.00	0.00

Taulukko 14: Muuttujien roolit Foleyn (1991) kirjassa esiintyvissä ohjelmissa (Sajaniemi, 2002b) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
stp	d1	1	0.65	2.50
stp	dn	0	0.00	0.00
stp	a2	0	0.00	0.00
stp	an	0	0.00	0.00
stp	ll	0	0.00	0.00
stp	ot	0	0.00	0.00
mrh		22	14.29	
mrh	in	22	14.29	100.00
mrh	ic	0	0.00	0.00
mrh	ae	0	0.00	0.00
mrh	dv	0	0.00	0.00
mrh	ot	0	0.00	0.00
gat		9	5.84	
gat	su	8	5.19	88.89
gat	de	0	0.00	0.00
gat	mu	1	0.65	11.11
gat	di	0	0.00	0.00
gat	ot	0	0.00	0.00
mwh		3	1.95	
mwh	ma	1	0.65	33.33
mwh	mi	1	0.65	33.33
mwh	cl	1	0.65	33.33
mwh	fa	0	0.00	0.00
mwh	fi	0	0.00	0.00
mwh	la	0	0.00	0.00
mwh	ot	0	0.00	0.00
one		0	0.00	
one	r1	0	0.00	0.00
one	rn	0	0.00	0.00
one	l1	0	0.00	0.00
one	ln	0	0.00	0.00

Taulukko 14: Muuttujien roolit Foleyn (1991) kirjassa esiintyvissä ohjelmissa (Sajaniemi, 2002b) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
one	ot	0	0.00	0.00
trn		24	15.58	
trn	fc	9	5.84	37.50
trn	rc	2	1.30	8.33
trn	fs	0	0.00	0.00
trn	rs	1	0.65	4.17
trn	fe	5	3.25	20.83
trn	re	5	3.25	20.83
trn	ot	2	1.30	8.33
fol		2	1.30	
fol	st	2	1.30	100.00
fol	ns	0	0.00	0.00
fol	ot	0	0.00	0.00
tmp		7	4.55	
tmp	sw	7	4.55	100.00
tmp	ot	0	0.00	0.00
oth		0	0.00	
oth	sc	0	0.00	0.00
oth	ar	0	0.00	0.00
oth	ot	0	0.00	0.00
yht		154	100	

Taulukko 15: Muuttujien roolit Jonesin (1982) kirjassa esiintyvissä ohjelmissa (Sajaniemi, 2002b).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fix		49	18.22	
fix	in	17	6.32	34.69
fix	ic	0	0.00	0.00
fix	co	13	4.83	26.53

Taulukko 15: Muuttujien roolit Jonesin (1982) kirjassa esiintyvissä ohjelmissa (Sajaniemi, 2002b) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fix	va	7	2.60	14.29
fix	pa	7	2.60	14.29
fix	dv	3	1.12	6.12
fix	ot	2	0.74	4.08
org		6	2.23	
org	so	4	1.49	66.67
org	re	0	0.00	0.00
org	ra	1	0.37	16.67
org	ot	1	0.37	16.67
stp		100	37.17	
stp	cn	25	9.29	25.00
stp	i1	43	15.99	43.00
stp	in	7	2.60	7.00
stp	nc	4	1.49	4.00
stp	d1	2	0.74	2.00
stp	dn	0	0.00	0.00
stp	a2	2	0.74	2.00
stp	an	3	1.12	3.00
stp	ll	6	2.23	6.00
stp	ot	8	2.97	8.00
mrh		44	16.36	
mrh	in	35	13.01	79.55
mrh	ic	1	0.37	2.27
mrh	ae	2	0.74	4.55
mrh	dv	6	2.23	13.64
mrh	ot	0	0.00	0.00
gat		18	6.69	
gat	su	10	3.72	55.56
gat	de	0	0.00	0.00
gat	mu	0	0.00	0.00
gat	di	1	0.37	5.56

Taulukko 15: Muuttujien roolit Jonesin (1982) kirjassa esiintyvissä ohjelmissa (Sajaniemi, 2002b) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
gat	ot	7	2.60	38.89
mwh		3	1.12	
mwh	ma	1	0.37	33.33
mwh	mi	0	0.00	0.00
mwh	cl	0	0.00	0.00
mwh	fa	0	0.00	0.00
mwh	fi	2	0.74	66.67
mwh	la	0	0.00	0.00
mwh	ot	0	0.00	0.00
one		5	1.86	
one	r1	1	0.37	20.00
one	rn	2	0.74	40.00
one	l1	0	0.00	0.00
one	ln	2	0.74	40.00
one	ot	0	0.00	0.00
trn		28	10.41	
trn	fc	8	2.97	28.57
trn	rc	1	0.37	3.57
trn	fs	1	0.37	3.57
trn	rs	3	1.12	10.71
trn	fe	4	1.49	14.29
trn	re	6	2.23	21.43
trn	ot	5	1.86	17.86
fol		9	3.35	
fol	st	7	2.60	77.78
fol	ns	2	0.74	22.22
fol	ot	0	0.00	0.00
tmp		2	0.74	
tmp	sw	2	0.74	100.00
tmp	ot	0	0.00	0.00

Taulukko 15: Muuttujien roolit Jonesin (1982) kirjassa esiintyvissä ohjelmissa (Sajaniemi, 2002b) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
oth		5	1.86	
oth	sc	2	0.74	40.00
oth	ar	3	1.12	60.00
oth	ot	0	0.00	0.00
yht		269	100	

Taulukko 16: Muuttujien roolit Peltomäen & Malmirakeen (1999) kirjassa esiintyvissä ohjelmissa (Byckling *et al.*, 2005).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fix		444	67.37	
fix	in	74	11.23	16.67
fix	ic	0	0.00	0.00
fix	co	54	8.19	12.16
fix	va	118	17.91	26.58
fix	pa	195	29.59	43.92
fix	dv	3	0.46	0.68
fix	ot	0	0.00	0.00
org		0	0.00	
org	so	0	0.00	0.00
org	re	0	0.00	0.00
org	ra	0	0.00	0.00
org	ot	0	0.00	0.00
stp		92	13.96	
stp	cn	27	4.10	29.35
stp	i1	43	6.53	46.74
stp	in	3	0.46	3.26
stp	nc	0	0.00	0.00
stp	d1	2	0.30	2.17

Taulukko 16: Muuttujien roolit Peltomäen & Malmirakeen (1999) kirjassa esiintyvissä ohjelmissa (Byckling *et al.*, 2005) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
stp	dn	1	0.15	1.09
stp	a2	1	0.15	1.09
stp	an	15	2.28	16.30
stp	ll	0	0.00	0.00
stp	ot	0	0.00	0.00
mrh		22	3.34	
mrh	in	17	2.58	77.27
mrh	ic	0	0.00	0.00
mrh	ae	3	0.46	13.64
mrh	dv	0	0.00	0.00
mrh	ot	2	0.30	9.09
gat		9	1.37	
gat	su	7	1.06	77.78
gat	de	0	0.00	0.00
gat	mu	2	0.30	22.22
gat	di	0	0.00	0.00
gat	ot	0	0.00	0.00
mwh		6	0.91	
mwh	ma	0	0.00	0.00
mwh	mi	0	0.00	0.00
mwh	cl	6	0.91	100.00
mwh	fa	0	0.00	0.00
mwh	fi	0	0.00	0.00
mwh	la	0	0.00	0.00
mwh	ot	0	0.00	0.00
one		20	3.03	
one	r1	4	0.61	20.00
one	rn	9	1.37	45.00
one	l1	1	0.15	5.00
one	ln	6	0.91	30.00

Taulukko 16: Muuttujien roolit Peltomäen & Malmirakeen (1999) kirjassa esiintyvissä ohjelmissa (Byckling *et al.*, 2005) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
one	ot	0	0.00	0.00
trn		54	8.19	
trn	fc	29	4.40	53.70
trn	rc	1	0.15	1.85
trn	fs	1	0.15	1.85
trn	rs	1	0.15	1.85
trn	fe	20	3.03	37.04
trn	re	2	0.30	3.70
trn	ot	0	0.00	0.00
fol		8	1.21	
fol	st	4	0.61	50.00
fol	ns	4	0.61	50.00
fol	ot	0	0.00	0.00
tmp		1	0.15	
tmp	sw	1	0.15	100.00
tmp	ot	0	0.00	0.00
oth		3	0.46	
oth	sc	0	0.00	0.00
oth	ar	2	0.30	66.67
oth	ot	1	0.15	33.33
yht		659	100	

Taulukko 17: Muuttujien roolit Wiklan (2003) kirjassa esiintyvissä ohjelmissa (Byckling *et al.*, 2005).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fix		132	43.71	
fix	in	24	7.95	18.18
fix	ic	0	0.00	0.00
fix	co	32	10.60	24.24
fix	va	18	5.96	13.64
fix	pa	57	18.87	43.18
fix	dv	1	0.33	0.76
fix	ot	0	0.00	0.00
org		3	0.99	
org	so	3	0.99	100.00
org	re	0	0.00	0.00
org	ra	0	0.00	0.00
org	ot	0	0.00	0.00
stp		63	20.86	
stp	cn	11	3.64	17.46
stp	i1	48	15.89	76.19
stp	in	0	0.00	0.00
stp	nc	1	0.33	1.59
stp	d1	2	0.66	3.17
stp	dn	0	0.00	0.00
stp	a2	0	0.00	0.00
stp	an	1	0.33	1.59
stp	ll	0	0.00	0.00
stp	ot	0	0.00	0.00
mrh		26	8.61	
mrh	in	23	7.62	88.46
mrh	ic	0	0.00	0.00
mrh	ae	1	0.33	3.85
mrh	dv	0	0.00	0.00
mrh	ot	2	0.66	7.69

Taulukko 17: Muuttujien roolit Wiklan (2003) kirjassa esiintyvissä ohjelmissa (Byckling *et al.*, 2005) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
gat		5	1.66	
gat	su	5	1.66	100.00
gat	de	0	0.00	0.00
gat	mu	0	0.00	0.00
gat	di	0	0.00	0.00
gat	ot	0	0.00	0.00
mwh		0	0.00	
mwh	ma	0	0.00	0.00
mwh	mi	0	0.00	0.00
mwh	cl	0	0.00	0.00
mwh	fa	0	0.00	0.00
mwh	fi	0	0.00	0.00
mwh	la	0	0.00	0.00
mwh	ot	0	0.00	0.00
one		10	3.31	
one	r1	8	2.65	80.00
one	rn	0	0.00	0.00
one	l1	2	0.66	20.00
one	ln	0	0.00	0.00
one	ot	0	0.00	0.00
trn		45	14.90	
trn	fc	39	12.91	86.67
trn	rc	2	0.66	4.44
trn	fs	0	0.00	0.00
trn	rs	0	0.00	0.00
trn	fe	3	0.99	6.67
trn	re	1	0.33	2.22
trn	ot	0	0.00	0.00
fol		2	0.66	
fol	st	0	0.00	0.00
fol	ns	2	0.66	100.00

Taulukko 17: Muuttujien roolit Wiklan (2003) kirjassa esiintyvissä ohjelmissa (Byckling *et al.*, 2005) (jatk.).

Rooli	*Alarooli*	*Lukumäärä*	*% kaikista*	*% roolista*
fol	ot	0	0.00	0.00
tmp		3	0.99	
tmp	sw	3	0.99	100.00
tmp	ot	0	0.00	0.00
oth		13	4.30	
oth	sc	0	0.00	0.00
oth	ar	8	2.65	61.54
oth	ot	5	1.66	38.46
yht		302	100	

Taulukko 18: Roolit ML-ohjelmissa (Kulikova, 2005). (1: sel-roolit käsiteltiin fix-rooleina, 2: mdf-rooleja ei huomioitu, 3: alaroolin luvut sisältyvät ylärooliin.)

Rooli	Ullman, 1998		Paulson, 1996		Hansen, 1999		Michaelson, 1995	
fix	99	26.5 %	379	27.6 %	322	40.0 %	315	31.0 %
fix_f[3]	14		31		26		62	
stp	61	16.4 %	175	12.8 %	77	9.6 %	168	16.5 %
stp_lst[3]	45		119		56		25	
mrh	75	20.1 %	302	22.0 %	125	15.5 %	169	16.6 %
mwh	3	0.8 %	6	0.4 %	0	0.0 %	5	0.5 %
sel[1]	8	2.1 %	49	3.6 %	22	2.7 %	48	4.7 %
mdf[2]	12	3.2 %	39	2.8 %	13	1.6 %	52	5.1 %
org	7	1.9 %	9	0.7 %	4	0.5 %	6	0.6 %
trn	69	18.5 %	180	13.1 %	131	16.3 %	144	14.2 %
gat	39	10.5 %	211	15.4 %	98	12.2 %	105	10.3 %
gat_lst[3]	15		157		53		63	
gat_ntr[3]	2		4		5		2	
one	0	0.0 %	9	0.7 %	9	1.1 %	4	0.4 %
fol	0	0.0 %	14	1.0 %	4	0.5 %	0	0.0 %
oth	0	0.0 %	0	0.0 %	1	0.1 %	0	0.0 %
Yhteensä	373 kpl	100 %	1373 kpl	100 %	806 kpl	100 %	1016 kpl	100 %

Tutkittaessa olivatko roolien esiintymistiheydet ohjelmistoasiantuntijoiden kirjoittamissa Java-ohjelmissa yhteneväiset, Megamek-, Hsqldb- ja Azureus-ohjelmien kolme pienintä roolia (org, fol, mwh) yhdistettiin (yhd; taulukko 19), jotta kaikissa ryhmissä olisi ollut ainakin viisi muuttujaa, ja χ^2-testi voitaisiin suorittaa. Vastaavasti toimittiin myös muissa vertailuissa, jotka on esitetty alla. Vertailujoukoksi luotiin kuvitteellinen tasainen jakauma, jossa jokaisella roolilla oli 50 esiintymää. Tutkimuksessa olleiden ohjelmistoasiantuntijoiden kirjoittamien Java-ohjelmien roolien esiintymistiheydet olivat keskenään tilastollisesti merkitsevästi erilaisia ($p <, 0001$).

Taulukko 19: Muuttujien roolien jakaumia ohjelmistoasiantuntijoiden kirjoittamissa Java-ohjelmissa. ($\chi^2 = 127.587, df = 8, p < .0001$.)

Rooli	Java-ohjelmat roolien jakauma	Keskimääräinen roolien jakauma
fix	130	50
stp	68	50
mrh	88	50
gat	14	50
one	34	50
trn	66	50
tmp	16	50
oth	24	50
yhd	10	50
Yhteensä	450 kpl	450 kpl

Testattaessa ohjelmistoasiantuntijoiden kirjoittamia Java-ohjelmia keskenään yhdistettiin jokaisesta ohjelmasta kuusi roolia (gat, tmp, oth, org, fol, mwh) yhdeksi (yhd; taulukko 20). Tulosten mukaan ohjelmistoasiantuntijoiden kirjoittamat Java-ohjelmat eivät eronneet toisistaan tilastollisesti merkitsevästi ($p =, 110$).

Vertailtaessa ohjelmistoasiantuntijoiden tekemissä aloitteleville ohjelmoijille suunnatuissa kahdessa kirjassa esiintyviä Java-ohjelmia keskenään koottiin molemmissa kirjoissa esiintyvien ohjelmien kuudesta roolista (gat, tmp, oth, org, fol, mwh) yksi ryhmä (yhd; Taulukko 21). Kirjoissa esiintyvien roolien jakaumat olivat keskenään tilastollisesti merkitsevästi erilaisia ($p <, 0001$).

Taulukko 20: Muuttujien roolien jakaumia ohjelmistoasiantuntijoiden kirjoittamissa Java-ohjelmissa. ($\chi^2 = 15.639, df = 10, p = .110$.)

Rooli	*Megamek*	*Hsqldb*	*Azureus*	*Yhteensä*
fix	38	49	43	130
stp	18	23	27	68
mrh	32	22	34	88
one	14	12	8	34
trn	31	21	14	66
yhd	17	23	24	64
Yhteensä	150 kpl	150 kpl	150 kpl	450 kpl

Taulukko 21: Muuttujien roolien jakaumia yksinkertaisissa Java-ohjelmissa. ($\chi^2 = 57.722, df = 5, p < .0001$.)

Rooli	*Peltomäki & Malmirae, 1999*	*Wikla, 2003*	*Yhteensä*
fix	444	132	576
stp	92	63	155
mrh	22	26	48
one	20	10	30
trn	54	45	99
yhd	27	26	53
Yhteensä	659 kpl	302 kpl	961 kpl

Aloitteleville ohjelmoijille suunnatuissa ja ohjelmistoasiantuntijoiden kirjoittamissa kolmessa kirjassa esiintyvissä Pascal-ohjelmissa yhdistettiin seitsemän roolia (gat, tmp, oth, org, fol, mwh, one) yhdeksi ryhmäksi (yhd; taulukko 22). Kirjoissa esiintyvien roolien jakaumat olivat tilastollisesti merkitsevästi erilaisia ($p <, 0001$).

Ohjelmistoasiantuntijoiden tekemissä aloitteleville ohjelmoijille suunnatuissa neljässä oppikirjassa esiintyviä ML-ohjelmia tutkittaessa yhdistettiin rooleista kuusi (tmp, oth, org, fol, mwh, one) omaksi ryhmäkseen (yhd; taulukko 23). Kirjoissa esiintyvien roolien jakaumat olivat keskenään tilastollisesti merkitsevästi erilaisia ($p <, 0001$).

Taulukko 22: Muuttujien roolien jakaumia yksinkertaisissa Pascal-ohjelmissa. ($\chi^2 = 30.140, df = 8, p < .0001$.)

Rooli	*Sajaniemi & Karjalainen, 1985*	*Foley, 1991*	*Jones, 1982*	*Yhteensä*
fix	54	40	49	143
stp	29	40	100	169
mrh	19	22	44	85
trn	18	24	28	70
yhd	16	28	48	92
Yhteensä	136 kpl	154 kpl	269 kpl	559 kpl

Taulukko 23: Muuttujien roolien jakaumia yksinkertaisissa ML-ohjelmissa. ($\chi^2 = 82.986, df = 15, p < .0001$.)

Rooli	*Ullman, 1998*	*Paulson, 1996*	*Hansen, 1999*	*Michaelson, 1995*	*Yhteensä*
fix	107	428	344	363	1242
stp	61	175	77	168	481
mrh	75	302	125	169	671
gat	39	211	98	105	453
trn	69	180	131	144	524
yhd	10	38	18	15	81
Yhteensä	361 kpl	1334 kpl	793 kpl	964 kpl	3452 kpl

Taulukossa 24 ovat χ^2-testin tulokset verrattaessa ohjelmistoasiantuntijoiden kirjoittamia Java-ohjelmia ohjelmistoasiantuntijoiden kirjoittamiin aloitteleville ohjelmoijille suunnattuihin oppikirjoihin ja niissä esiintyviin Java-ohjelmiin. Ohjelmista yhdistettiin kuusi roolia (gat, tmp, oth, org, fol, mwh) yhden ryhmän alle (yhd). Molemmissa aloitteleville ohjelmoijille suunnatuissa kirjoissa roolien jakaumat erosivat merkitsevästi ohjelmistoasiantuntijoiden kirjoittamista Java-ohjelmista ($p <, 0001$).

Taulukossa 25 ovat χ^2-testin tulokset verrattaessa ohjelmistoasiantuntijoiden kirjoittamia Java-ohjelmia ohjelmistoasiantuntijoiden kirjoittamiin aloitteleville ohjelmoijille suunnattuihin oppikirjoihin ja niissä esiintyviin Pascal-ohjelmiin. Ohjelmista yhdistettiin seitsemän roolia (gat, tmp, oth, org, fol, mwh, one) yhden ryhmän alle (yhd). Pascal-ohjelmat, jotka esiintyvät Jonesin (1982) kirjassa, olivat roolien jakaumien suh-

teen tilastollisesti merkitsevästi erilaisia verrattuna ohjelmistoasiantuntijoiden kirjoittamiin Java-ohjelmiin ($p <, 0001$). Sajaniemen & Karjalaisen (1985) kirjassa esiintyvät ohjelmat olivat myös roolien jakaumien suhteen tilastollisesti merkitsevästi erilaisia verrattuna ohjelmistoasiantuntijoiden kirjoittamiin Java-ohjelmiin ($p =, 009$). Myös Foleyn (1991) kirjassa esiintyvien roolien jakaumat erosivat tilastollisesti merkitsevästi ohjelmistoasiantuntijoiden kirjoittamista Java-ohjelmista ($p =, 034$).

Taulukko 24: Roolien jakaumia ohjelmistoasiantuntijoiden kirjoittamissa Java-ohjelmissa[1] verrattuna yksinkertaisiin Java-ohjelmiin[2,3]. (2: $\chi^2 = 202.653, df = 5, p < .0001$, 3: $\chi^2 = 39.433, df = 5, p < .0001$.)

Rooli	*Ohjelmistoasiantuntijat[1]*	*Peltomäki & Malmirae, 1999[2]*	*Wikla, 2003[3]*
fix	130	444	132
stp	68	92	63
mrh	88	22	26
one	34	20	10
trn	66	54	45
yhd	64	27	26
Yhteensä	450 kpl	659 kpl	302 kpl

Taulukko 25: Roolien jakaumia ohjelmistoasiantuntijoiden kirjoittamissa Java-ohjelmissa[1] verrattuna yksinkertaisiin Pascal-ohjelmiin[2,3,4]. (2: $\chi^2 = 13.643, df = 4, p = .009$, 3: $\chi^2 = 10.444, df = 4, p = .034$, 4: $\chi^2 = 47.366, df = 4, p < .0001$.)

Rooli	Ohjelmisto-asiantuntijat [1]	Sajaniemi & Karjalainen, 1985[2]	Foley, 1991[3]	Jones, 1982[4]
fix	130	54	40	49
stp	68	29	40	100
mrh	88	19	22	44
trn	66	18	24	28
yhd	98	16	28	48
Yhteensä	450 kpl	136 kpl	154 kpl	269 kpl

Taulukossa 26 ovat χ^2-testin tulokset verrattaessa ohjelmistoasiantuntijoiden kirjoittamia Java-ohjelmia ohjelmistoasiantuntijoiden kirjoittamiin aloitteleville ohjelmoijille suunnattuihin ML-oppikirjoihin ja niissä esiintyviin muuttujien rooleihin. Kaikista ohjelmista on yhdistetty kuusi roolia (tmp, oth, org, fol, mwh, one) yhden ryhmän alle (yhd). Kaikki neljä aloitteleville ohjelmoijille suunnattua oppikirjaa olivat tilastollisesti merkitsevästi erilaisia niissä esiintyvien roolien jakaumien suhteen verrattuna ohjelmistoasiantuntijoiden kirjoittamiin Java-ohjelmiin ($p <, 0001$).

Taulukko 26: Roolien jakaumia ohjelmistoasiantuntijoiden kirjoittamissa Java-ohjelmissa[1] verrattuna yksinkertaisiin ML-ohjelmiin[2,3,4,5]. (2: $\chi^2 = 64.776, df = 5, p < .0001$, 3: $\chi^2 = 170.068, df = 5, p < .0001$, 4: $\chi^2 = 147.323, df = 5, p < .0001$, 5: $\chi^2 = 158.815, df = 5, p < .0001$.)

Rooli	Ohjelmisto-asiantuntijat[1]	Ullman, 1998[2]	*Paulson, 1996*[3]	*Hansen, 1999*[4]	*Michaelson,* 1995[5]
fix	130	107	428	344	363
stp	68	61	175	77	168
mrh	88	75	302	125	169
gat	14	39	211	98	105
trn	66	69	180	131	144
yhd	84	10	38	18	15
Yht.	450 kpl	361 kpl	1334 kpl	793 kpl	964 kpl

5 Tulosten pohdinta

Kohdassa 5.1 tarkastellaan saatuja tuloksia ja pohditaan sitä, miksi tulokset ovat juuri tällaisia. Tulosten tarkastelun jälkeen kohdassa 5.2 mietitään, ovatko tulokset kelvollisia lähdemateriaalin valinnan, laajuuden ja muiden tulosten kelpoisuuteen vaikuttavien seikkojen suhteen.

5.1 Tulosten tarkastelu

Tulosten perusteella ei voida tehdä varmoja yleistyksiä ohjelmistoasiantuntijoiden kirjoittamien Java-ohjelmien suhteen, mutta tutkittujen kolmen ohjelman perusteella on kuitenkin nähtävissä, että ne ovat keskenään joiltakin osin samankaltaisia. Ohjelmien väliset erot eivät ole tulosten perusteella tilastollisesti merkitseviä, vaikka ohjelmat suorittavatkin toisistaan selkeästi eroavia tehtäviä ja niiden kohdealueet eroavat toisistaan. Roolitetuista ohjelmistoasiantuntijoiden kirjoittamista ohjelmista on helppo huomata niiden samankaltaisuus käytettyyn paradigmaan liittyvissä yksityiskohdissa kuten silmukkarakenteissa, olioiden välisessä kommunikoinnissa, boolean-arvojen, muuttujien ja vakioiden käytössä sekä kielen syntaksiin kuuluvien tiedon ja funktion yhdistävien olioiden kuten Java-Stringin käsittelyssä. Ohjelmien välillä on kuitenkin selkeitä eroja niiden suorittamissa tehtävissä sekä kohdealueissa. Megamekin asiakas-palvelin-ohjelma suorittaa muun muassa pelitilanteen luomisen, pelitilanteen tietojen ylläpidon ja raportoinnin sekä pelitilanteen tallentamisen, kun taas Hsqldb:stä valittu osa on enimmäkseen tietokantakyselyiden jäsentämistä ja oikeellisuuden varmistamista eli käytännössä sql-komentoja sisältävien tekstitiedostojen enkapsulointia sekä lopulta niistä saatujen ja muokattujen tietojen syöttämistä jdcb-yhteydelle. Azureus-ohjelmasta valitut osat puolestaan käsittelevät objekteja joko lista- tai taulukkomuodossa käyttäjän suorittamien hiiri- ja näppäimistökomentojen perusteella. Jos näin erilaisia tehtäviä suorittavista ohjelmista ei löydy tilastollisesti merkitseviä eroja roolien esiintymisessä, on todennäköistä, että jos muita satunnaisesti valittuja Java-ohjelmia verrattaisiin tutkimuksestani saatuihin tuloksiin, olisivat tulokset monen ohjelman osalta samansuuntaisia muuttujien roolien suhteen. Tutkielmani perusteella saatuja tuloksia ei voi vielä yleistää, sillä erilaisia ohjelmia ja ohjelmoijia on valtavasti.

Tulosten perusteella voidaan kuitenkin todeta, että oppikirjoissa esiintyvät ohjelmistoasiantuntijoiden tekemät noviiseille suunnatut ML-, Pascal- ja Java-ohjelmat ovat

suurella todennäköisyydellä erilaisia verrattuna ohjelmistoasiantuntijoiden kirjoittamiin Java-ohjelmiin. Noviiseille suunnattujen oppikirjoissa esiintyvien ohjelmien ja ohjelmistoasiantuntijoiden kirjoittamien Java-ohjelmien väliset erot on helppo huomata, sillä oppikirjoissa esiintyvät esimerkit ovat hyvin lyhyitä verrattuna ohjelmistoasiantuntijoiden kirjoittamiin tuotantokäyttöön tarkoitettuihin ohjelmiin ja suorittavat harvoin mitään muuta kuin erittäin vähämerkityksellisiä toimenpiteitä. Oppikirjoissa esiintyvät Java- ja Pascal-esimerkit voivat kuitenkin olla selkeästi lähempänä ohjelmistoasiantuntijoiden kirjoittamia Java-ohjelmia kuin ML-oppikirjojen esimerkit, koska ne perustuvat pohjimmiltaan samaan paradigmaan, vaikka Javassa onkin mukana olioaspekti. Ohjelmistoasiantuntijoiden kirjoittamissa Java-ohjelmissa ja ohjelmistoasiantuntijoiden noviiseille suunnatuissa ML-oppikirjojen ohjelmissa on toki samankaltaisuuksiakin, mutta nämä yhteneväisyydet ovat pääasiassa niitä, joita löytyy lähes kaikista ohjelmista kuten yksinkertaisia silmukkarakenteita ja vakioarvojen asettamisia sekä muita ohjelmointikielistä löytyviä perusrakenteiden yhtäläisyyksiä. Monia näistä on kuitenkin vaikea löytää ML-ohjelmista, olivatpa ne tutkielmassa esiintyviä lyhyitä oppikirjaesimerkkejä tai ohjelmistoasiantuntijoiden tekemiä ohjelmia. Tulosten perusteella voidaan sanoa, että ohjelmistoasiantuntijoiden kirjoittamat noviiseille suunnatut oppikirjaesimerkit eroavat tuotantokäyttöön tarkoitetuista ohjelmista muuttujien roolien esiintymisen suhteen.

Jos tuloksia tarkastellaan tilastollisten testien ulkopuolella eli verrataan eri muuttujaryhmien prosenttiosuuksia ja ryhmien kokoja keskenään, voidaan luoda yleiskuva kunkin muuttujan roolin tärkeydestä, mitä χ^2-testi ei kerro. Taulukkoon 8 on koottu tulososiossa läpikäydyt muuttujien prosenttiosuuksien vaihteluvälit ohjelmittain, taulukosta 7 löytyvät samat luvut ohjelmatyypeittäin. Jokaisessa ohjelmassa ja lähes jokaisessa oppikirjassa suurin rooliryhmä oli *kiintoarvot*, Peltomäen & Malmirakeen (1999) Java-oppaassa näiden osuus oli erityisen korkea, 67,4 %. Edellämainitusta poikkesi ainoastaan Jonesin (1982) kirjoittama noviiseille suunnattu Pascal-kielen oppikirja, ja siinäkin *kiintoarvot* oli toiseksi suurin ryhmä. Tulosten perusteella voidaan siis todeta, että *kiintoarvo* on selkeästi yleisin muuttujien rooli kaikissa käsitellyissä ohjelmointikielissä. Seuraavat kolme suurinta ryhmää olivat *askeltajat*, *tuoreimman säilyttäjät* ja *muuntajat*. Poikkeuksena tästä oli yksi ML-kielen oppikirja (Paulson, 1996), jossa kolmanneksi yleisin rooli oli *kokoojat* vain 2,2 % erolla ennen *muuntajia*. Jos *muuntajia* käsittelee uusimpien roolitussääntöjen (taulukko 5) mukaisesti, kolmen suurimman ryhmän joukossa *muuntajien* tilalla olisi tällöin *yksisuuntaiset liput* (taulukko 9), ja kuudenneksi suurimmaksi ryhmäksi nousisi uusi rooli *säiliöt*. Lukumääräiset ja prosentuaali-

set erot *askeltaja*-roolin esiintymisessä eri ohjelmien ja ohjelmatyyppien välillä selittynevät ohjelmissa esiintyvien ohjaussilmukoiden määrällä. Yleensä myös *tuoreimman säilyttäjien* määrä kasvaa samalla, kun silmukoiden määrä kasvaa, sillä niitä käytetään paljon silmukoiden sisällä. Tämä ei kuitenkaan käynyt ilmi Peltomäen & Malmirakeen (1999) eikä Wiklan (2003) teoksista, joissa esiintyvissä ohjelmissa ei oltu usein käytetty *tuoreimman säilyttäjä* -roolia silmukoiden sisällä. Pienimpiä ryhmiä olivat *sopivimman säilyttäjät*, *seuraajat*, *tilapäissäilöt* sekä *muut*, joista ainoastaan *muut*-ryhmä oli Azureus- ja Hsqldb-ohjelmissa hieman suurempi johtuen niissä esiintyvistä roolitussääntöjen version 2.0 mukaisista *säiliö*-tyyppisistä taulukoista, joita ei tutkielmassa osattu sijoittaa roolitussääntöjen version 1.2 mukaisiin muihin ryhmiin. *Muut*-ryhmän pienuus tukee tutkielmassa käytettyjen roolien sopivuutta kaikkien muuttujien roolittamiseen. *Tilapäissäilö*-ryhmästä suurin osa muuttujista sisältyi *muuntajat*-ryhmään, *sopivimman säilyttäjien* ja *seuraajien* vähyys taas johtui muun muassa listojen toteutuksesta Java-kielessä sekä näiden vähäisestä esiintymisestä tutkituissa oppikirjoissa. ML-oppikirjojen ohjelmissa erottui *kokoojien* suhteellisen suuri määrä.

Roolitussääntöjen versio 2.0 (taulukko 5) näyttäisi soveltuvan paremmin kuin roolitussääntöjen versio 1.2 (taulukko 4) ainakin ohjelmistoasiantuntijoiden kirjoittamien Java-ohjelmien muuttujien roolien analysointiin. Käytettäessä roolitussääntöjen versiota 2.0 Azureus-ohjelman *muut*-ryhmässä ei ollut enää yhtään esiintymää, kun roolitussääntöjen versiolla 1.2 näitä oli vielä yhdeksän (6,0 %). Hsqldb-ohjelman *muut*-ryhmän koko pieneni samoin 15 esiintymästä (10,0 %) kolmeen (2,0 %). Myös ohjelmistoasiantuntijoiden kirjoittamiin noviiseille suunnattuihin oppikirjoihin ja niissä esiintyvin Java- ja Pascal-ohjelmiin uusimmat version 2.0 mukaiset roolitussäännöt näyttäisivät sopivan paremmin kuin vanhat version 1.2 mukaiset roolitussäännöt. Näissä kirjoissa *muut*-ryhmän koko pieneni seuraavasti: 1,1 % (Foley, 1991), 0,3 % (Peltomäki & Malmirae, 1999) ja 2,6 % (Wikla, 2003). Sajaniemen & Karjalaisen (1985) ja Jonesin (1982) kirjoissa *muut*-ryhmän koko ei muuttunut, joten on mahdollista, että roolitussääntöjen muutos versiosta 1.2 versioon 2.0 vaikuttaa enemmän Java-kielellä kuin Pascal-kielellä kirjoitettujen ohjelmien roolittamiseen.

Vaatisi paljon lisätyötä selvittää, miksi ohjelmistoasiantuntijoiden kirjoittamat noviiseille suunnatut oppikirjaesimerkit ovat Java-, Pascal- ja ML-ohjelmistokielien sisällä muuttujien roolien suhteen niin erilaisia. Syynä voivat olla erot oppikirjojen lähestymistavoissa tai aihealueissa ja siten jonkin yksittäisen muuttujan roolin korostumisessa tai mahdollisesti kirjoittajien erilaisissa ohjelmointitavoissa. Ohjelmat kuitenkin perus-

tuvat samaan paradigmaan, joten ainakaan tästä erot eivät voi johtua. Tulosten mukaan on kuitenkin perusteltua sanoa, että oppikirjoissa esiintyvät ohjelmistoasiantuntijoiden kirjoittamat ohjelmat eroavat toisistaan muuttujien roolien jakauman suhteen, ja suurella todennäköisyydellä eroja voi olla myös ainakin eri ohjelmistoasiantuntijoiden kirjoittamien teosten välillä. Tutkimuksen tulokset tukevat jo aikaisemmin saatuja selvityksiä, joiden mukaan ohjelmistoasiantuntijoiden ja noviisien kirjoittamien ohjelmien väliset erot ovat enimmäkseen vain käytettyjen roolien jakaumissa, itse roolit pysyvät muuttumattomina (Sajaniemi 2002b; Sajaniemi & Navarro Prieto, 2005).

5.2 Tulosten kelpoisuus

Korkealaatuisessa tutkimuksessa valitun lähdemateriaalin tulisi vastata mahdollisimman hyvin tarkasteltavaa kohdejoukkoa. Lähdemateriaalin rajoittuessa ohjelmistoasiantuntijoiden kirjoittamien Java-ohjelmien osalta kolmeen ohjelmaan ja niissä oleviin vajaaseen 30 000 ohjelmakoodirivin sekä 3 000 muuttujan tarkasteluun on lähdemateriaalin määrä melko pieni, koska esimerkiksi Azureus-ohjelman lähdekoodin kokonaislaajuus on suurempi kuin 22 miljoonaa tavua. Jos jokaisen koodirivin olettaisi olevan kooltaan 50 merkkiä, koko ohjelmassa olisi yli 400 000 riviä ohjelmakoodia ja tutkitun ohjelmanosan muuttujien esiintymistiheyden perusteella yli 50 000 muuttujaa (11,4 muuttujaa 100 ohjelmariviä kohden). Tällöin Azureus-ohjelmasta tutkittavan osuuden 832 muuttujaa vastaisivat tästä alle kahta prosenttia, ja roolitettujen 150 muuttujan osuus näistä olisi vain noin kolme promillea. On myös hyvin vaikeaa todentaa, edustavatko valitut ohjelmat kattavasti kaikkia kirjoitettuja Java-ohjelmia.

Roolien kelpoisuuden tarkastelu ei aiheuta tutkimuksen sisällä muutoksia, mutta jos käytetyt roolit eivät ole universaalisti hyväksyttävissä, tutkimuksissa tehtyjä roolituksia ei voida verrata uusien erilaisten roolitussääntöjen perusteella tehtyihin myöhempiin tutkimustuloksiin. Tässä tutkimuksessa käytettiin version 1.2 mukaisia roolitussääntöjä (Sajaniemi, 2004), mutta saadut tulokset on helppo muuntaa uusimman version 2.0 roolitussäännöstön mukaisiksi (Sajaniemi *et al.*, 2006) muuttamalla *muuntaja*-rooli *tilapäissäilö*-rooliksi ja *muut*-roolin alarooli (oth ar) *säiliöksi*. Suuremmat muutokset roolitussäännöissä voivat tulevaisuudessa vaikeuttaa tulosten tulkintaa ja tehdä vertailun uusilla roolitussäännöillä saatuihin tuloksiin mahdottomaksi. Lukumääräisesti pieniksi jääneiden ryhmien yhdistäminen χ^2-testin suorittamisen mahdollistamiseksi voi myös vaikuttaa ohjelmien välisten tilastollisten erojen havaitsemiseen ja näin

vääristää loppupäätelmiä. Aikaisemmin suoritettu ML-ohjelmien roolitus (Kulikova, 2005) ja näiden roolien sekä roolituksien erilaisuus (taulukko 18) verrattuna tämän tutkimuksen tuloksiin (taulukot 6, 11 ja 12) sekä aikaisemmin Java- ja Pascal-ohjelmista tehtyihin roolituksiin (taulukot 13, 14, 15, 16, 17; Sajaniemi *et al.*, 2006) voivat lisätä virheellisten päätelmien tekemisen riskiä.

Tehdyn roolitustyön tarkkuuden vaihtelu roolitettujen ohjelmien välillä on myös otettava huomioon. Ohjelmistoasiantuntijoiden kirjoittamat ja noviiseille suunnatut Pascal- ja Java-ohjelmat roolitettiin pienemmällä virhemarginaalilla kuin muut. Pascal- ja Java-ohjelmat roolitettiin kahden tutkijan yhteistyönä siten, että molemmat tutkijat roolittivat ensin kaikki ohjelmat ja keskustelivat lopuksi roolituksien välisistä eroista kunnes olivat samaa mieltä lopputuloksesta (Sajaniemi *et al.*, 2006). ML-ohjelmien roolitustyön teki yksittäinen henkilö (Kulikova, 2005), jolloin virheellisten roolituksien lukumäärä on todennäköisesti suurempi kuin, jos kyseessä olisivat olleet kaksi toisistaan riippumatonta roolittajaa. Tämän tutkimuksen tulokset ovat myös yksittäisen henkilön työn lopputulos, ja vaikka materiaali käytiin kahdesti lävitse, on virhemarginaali suurempi kuin, jos kyseessä olisivat kaksi toistensa virheitä korjaavaa roolittajaa. Lisäksi roolitussääntöjen version 1.1 perusteella tehtyjen ohjelmistoasiantuntijoiden noviiseille suunnattujen Java- ja Pascal-oppirjojen muuttujien roolituksien muuntaminen roolitussääntöjen 2.0 mukaisiksi tehtiin ilman alkuperäisten muuttujien roolien tarkistamista.

Ohjelmistoasiantuntijoiden tekemien noviiseille suunnattujen ohjelmien tarkastelu aloittelevien ohjelmoijien tekeminä ohjelmina vaikuttaa suuresti tulosten soveltamiseen aloittelevien ohjelmoijien tekemien ohjelmien suhteen. Vaikka tulokset tukevat aikaisemmin saatuja johtopäätöksiä muuttujien roolien muuttumattomuudesta erilaisten ohjelmoijien kesken (Sajaniemi, 2002b; Sajaniemi & Navarro Prieto, 2005), on roolien jakaumien muuttumattomuuden todistaminen vielä avoin aloittelevien ohjelmoijien osalta johtuen aikaisempien ja tämän tutkimuksen käyttämän lähdemateriaalin vajavaisuudesta. Ohjelmistoasiantuntijoiden kirjoittamat yksinkertaisetkaan ohjelmat eivät ole verrattavissa aloittelevien ohjelmoijien ohjelmiin, eikä aloittelevien ohjelmoijien ymmärrys ohjelmista ole lähelläkään ohjelmistoasiantuntijoiden tasoa (McKeithen *et al.*, 1981).

6 Yhteenveto

Eri ohjelmointiparadigmoihin perustuvissa ohjelmissa muuttujien esiintymiseen voidaan liittää roolit käyttämällä pientä ryhmää stereotyyppisiä malleja, joita kutsutaan muuttujien rooleiksi. Tutkielmassa selvitettiin, kuinka nykyiset muuttujien roolit toimivat Java-kielellä kirjoitetuissa ohjelmissa, mikä on kunkin roolin esiintymistiheys, lukumäärä ja prosentuaalinen osuus kaikista rooleista, ja olisiko olemassaoleviin muuttujien rooleihin tehtävä lisäyksiä, kun kyseessä ovat ohjelmistoasiantuntijoiden tekemät Java-ohjelmat.

Tutkielmassa tarkasteltiin kolmea erilaista ohjelmaa, joiden muuttujat roolitettiin aikaisemmissa tutkimuksissa luotujen yleisten roolitussääntöjen perusteella sekä tutkimuksessa määritettyjen olioiden käsittelyyn kohdistuvien erikoissääntöjen perusteella. Saatuja tuloksia verrattiin myös jo aikaisemmin roolitettuihin aloitteleville ohjelmoijille suunnatuista mutta ohjelmistosiantuntijoiden kirjoittamista Java-, Pascal- ja ML-ohjelmista saatuihin tuloksiin.

Ohjelmistoasiantuntijoiden kirjoittamien Java-ohjelmien kesken ei muuttujien roolien jakaumissa ollut tilastollisesti merkitseviä eroja. Roolien jakaumat erosivat toisistaan tilastollisesti merkitsevästi ohjelmistoasiantuntijoiden aloitteleville ohjelmoijille suunnattujen ohjelmien välillä sekä näiden ja ohjelmistoasiantuntijoiden kirjoittamien Java-ohjelmien välillä. Muuttujien rooleista selkeästi suurin ryhmä oli *kiintoarvot*, muita yleisiä rooleja olivat *askeltajat* sekä *tuoreimman säilyttäjät*, ja pienimpiä rooliryhmiä olivat *sopivimman säilyttäjät*, *seuraajat* sekä *tilapäissäilöt*. Yhdestä ohjelmasta tutkitut parametrit olivat kaikki *kiintoarvoja*.

Tutkimuksen tulosten perusteella rooleihin ei ole tarvetta tehdä lisäyksiä, sillä nykyisin käytössäoleva muuttujien roolitusmalli soveltuu hyvin myös ohjelmistoasiantuntijoiden kirjoittamiin Java-ohjelmiin. Tutkituista 450 muuttujasta ainoastaan kolme jäi ryhmään *muut*, eli niitä ei voitu sijoittaa mihinkään rooliin.

Saman aihepiirin sisällä mahdollisesti tehtävissä jatkotutkimuksissa voitaisiin lisätä tutkittavan aineiston määrää ja mahdollisuuksien mukaan käyttää roolituksessa useampia henkilöitä sekä kehitteillä olevia muuttujien roolien roolituksen apuvälineitä.

Kiitokset

Professori Jorma Sajaniemi antoi minulle hienon mahdollisuuden tutustua tutkielmassani uuteen ja merkitykselliseen tutkimusaiheeseen eli muuttujien rooleihin. Aihealuetta on Sajaniemen tekemän työn lisäksi tutkittu aikaisemmin vähän varsinkin ohjelmistoasiantuntijoiden kirjoittamien laajojen ohjelmien suhteen, ja tutkielmani pyrkiikin osaltaan täyttämään tämän puutteen. Prosessi oli pitkä ja raskas eikä olisi varmasti valmistunut ilman Anne-Mari Mustosen ja Petteri Niemisen tukea ja kannustusta, eikä Joensuun yliopiston tarjoama työtila ainakaan haitannut työn edistymistä. Lisäksi haluan kiittää Arto Tuppuraista vertaistuesta; rotan on nopeampi paeta uppoavasta laivasta, kun vieressä on toinen, joka kertoo veden tulvivan jo kannelle :-)

Viitteet

Ben-Ari M., Sajaniemi J. (2004) Roles of variables as seen by CS educators. *Proceedings of the Ninth Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE´04)* (toim. Boyle R., Clark M., Kumar M.), ACM Press, New York, Yhdysvallat, 52–56.

Byckling P., Gerdt P., Sajaniemi J. (2005) Roles of variables in object-oriented programming. *Companion to the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA 2005)* (toim. Johnson R., Gabriel R. P.), ACM Press, New York, Yhdysvallat, 350–355.

Byers J. A. (1991) BASIC algorithms for random sampling and treatment randomization. *Computers in Biology and Medicine* **21**, 69–77.

Davis M. (2003) *Tietokoneen esihistoria Leibnizista Turingiin*. Art House, Helsinki, Suomi.

Détienne F. (2002) *Software Design–Cognitive Aspects*. Springer-Verlag, Lontoo, Iso-Britannia.

Ehrlich K., Soloway E. (1984) An empirical investigation of the tacit plan knowledge in programming. *Human Factors in Computer Systems* (toim. Thomas J. C., Schneider M. L.), Ablex Publishing Company, Norwood, New Jersey, Yhdysvallat, 113–133.

Felleisen M., Findler R. B., Flatt M., Krishnamurthi S. (2004) The Teach-32 Scheme! project: Computing and programming for every student. *Computer Science Education* **14**, 55–77.

Foley R.W. (1991) *Introduction to Programming Principles Using Turbo Pascal*. Chapman & Hall, Lontoo, Iso-Britannia.

Green T. R. G., Cornah A. J. (1984) The programmer's torch. *Human-Computer Interaction - INTER-ACT´84* (toim. Shackel B.), Elsevier Science Publishers, North Holland, Alankomaat, 397–402.

Hansen M. R., Rischel H. (1999) *Introduction to Programming Using SML*. Addison-Wesley, Harlow, Iso-Britannia.

Johnson-Laird P. N. (1987) *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Cambridge University Press, Massachusetts, Yhdysvallat.

Jones W. B. (1982) *Programming Concepts – A Second Course*. Prentice-Hall, Englewood Cliffs, New Jersey, Yhdysvallat.

Kuittinen M., Sajaniemi J. (2004) Teaching roles of variables in elementary programming courses. *Proceedings of the Ninth Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE´04)* (toim. Boyle R., Clark M., Kumar M.), ACM Press, New York, Yhdysvallat, 57–61.

Kulikova Y. (2005) *Roles of Variables in Functional Programming*. Pro gradu -tutkielma, Joensuun yliopisto, Joensuu, Suomi.

McKeithen K. B., Reitman J. S., Rueter H. H., Hirtle S. C. (1981) Knowledge organization and skill differences in computer programmer. *Cognitive Psychology* **13**, 307–325.

Michaelson G. (1995) *Elementary Standard ML*. UCL Press, Lontoo, Iso-Britannia.

Paulson L. C. (1996) *ML for the Working Programmer*, 2nd edition. Cambridge University Press, Cambridge, Iso-Britannia.

Peltomäki J., Malmirae P. (1999) *Java-ohjelmoinnin peruskirja*. Teknolit Oy, Jyväskylä, Suomi.

Rist R. S. (1989) Schema creation in programming. *Cognitive Science* **13**, 389–414.

Sajaniemi J. (2002a) A new approach to variable visualization: Roles as visualization objects. *Proceedings of the Second Program Visualization Workshop* (toim. Ben-Ari M.), University of Aarhus, Århus, Tanska, 75–83.

Sajaniemi J. (2002b) 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)* (toim. Werner B.), IEEE Computer Society, Arlington, Virginia, Yhdysvallat, 37–39.

Sajaniemi J. (2003) *BASICS*. WWW-sivusto, `http://cs.joensuu.fi/~saja/var_role_analyses/BASICS.tar` (30.05.2008).

Sajaniemi J. (2004) *Muuttujien roolituksessa käytetyt tarkennetut roolit*. WWW-sivusto, `http://cs.joensuu.fi/~saja/var_role_analyses/C_roles_proc_1.2` (27.05.2008).

Sajaniemi J., Karjalainen M. (1985) *Suppea johdatus Pascal-ohjelmointiin*. Joensuun yliopisto, Epsilon ry, Joensuu, Suomi.

Sajaniemi J., Navarro Prieto R. (2005) Roles of variables in experts' programming knowledge. *Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group (PPIG 2005)* (toim. Romero P., Good J., Bryant S., Chaparro E. A.), University of Sussex, Brighton, Iso-Britannia, 145–159.

Sajaniemi J., Ben-Ari M., Byckling P., Gerdt P., Kulikova Y. (2006) Roles of variables in three programming paradigms. *Computer Science Education* **16**, 261–279.

Sebesta R. W. (2003) *Concepts of Programming Languages*, 5th edition. Addison-Wesley, Boston, Massachusetts, Yhdysvallat.

SourceForge (2008) *sourceforge.net*. WWW-sivusto, `http://sourceforge.net` (25.03.2008).

Ullman J. D. (1998) *Elements of ML Programming*. Prentice-Hall, Upper Saddle River, New Jersey, Yhdysvallat.

Wikla A. (2003) *Ohjelmoinnin perusteet Java-kielellä*. OtaDATA, Espoo, Suomi.