A Study on Applying Roles of Variables in Introductory Programming

Pauli Byckling
University of Joensuu
Department of Computer Science and Statistics
Joensuu, Finland
pauli.byckling@cs.joensuu.fi

Jorma Sajaniemi
University of Joensuu
Department of Computer Science and Statistics
Joensuu, Finland
jorma.sajaniemi@cs.joensuu.fi

Abstract

Expert programmers possess programming knowledge, which is language independent and abstract. Still, programming is mostly taught only via constructs of a programming language and explicit teaching of programming knowledge is often disregarded. Experts’ high level programming knowledge can, however, be explicitly taught in an introductory programming course. This paper reports the results of a study in which the effects of teaching roles of variables to novices in an elementary programming course were examined. In the course expert programming knowledge was taught explicitly with the concept of roles and by using PlanAni, a role-based animator, which elaborates role knowledge and the concept of a variable in general. The study replicates methodologically our earlier study conducted in the same course a year earlier when roles were not used in the teaching. In this paper we discuss also the differences between the two student groups taught differently. The overall results show that students who were taught with roles learned high level programming plans faster and were able to use some complex programming plans better than students who were taught only with traditional methods.

1. Introduction

Expert programmers are known to possess high level programming knowledge, which is language independent and abstract and which has to be gained one way or another along the way to expertise [5]. However, in teaching programming teachers still tend to prefer teaching via syntax and language constructs of a programming language [8]. Thus, what is really taught in programming courses is mainly programming language knowledge instead of more general and more important programming knowledge.

When considering the learning process in an elementary programming course, learning the syntax of a programming language is not usually hard for a novice student. According to Robins et al. [8], virtually every study has shown that students have no trouble generating syntactically valid statements once they understand what is needed. The difficulty in learning programming is to learn to know where and how to combine statements to generate required results, i.e., to apply programming knowledge. Learning this skill takes usually the most time and effort—still, it is rarely directly supported by the teacher or the materials. Moreover, some teachers tend to prefer syntactic knowledge even when assessing students’ performance; e.g., in a program comprehension task of a classroom experiment [13] teachers tended to give better grades for detailed surface structure summaries than for answers revealing deep understanding.

Earlier studies have shown that explicit teaching of high-level programming knowledge to novices is indeed possible. In a classroom experiment, explicit teaching of roles of variables [9, 10] resulted in better programming skills [4] and in better mental models of programs [13]. The concept of roles of variables consists of a comprehensive, yet compact, set of characterizations of variables that can be used for teaching programming. Roles have been applied to object-oriented, procedural and functional programming [11], and they have been found to be a part of experts’ programming knowledge [14]. Table 1 lists eleven roles that suffice to cover almost all variables in elementary programming [11].

When using roles in introductory teaching, role knowledge can be further advanced by role-based program visualization and animation. PlanAni [12] is a role-based program animator, which elaborates high-level programming knowledge by using role images for visualizing variables and role-based animation for visualizing operations. A role image—a visualization used for all variables of the role—gives clues on how the successive values of the variable relate to each other and to other variables, e.g., the role image for fixed value is a stone giving the impression of immutability. In the very first phase of learning to program, PlanAni illustrates also the concept of a variable in general. Visualizing
Table 1. Roles of variables in novice-level object-oriented, procedural and functional programming [11].

<table>
<thead>
<tr>
<th>Role</th>
<th>Informal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed value (FIX)</td>
<td>A data item that does not get a new proper value after its initialization.</td>
</tr>
<tr>
<td>Stepper (STP)</td>
<td>A data item stepping through a systematic, predictable succession of values.</td>
</tr>
<tr>
<td>Most-recent holder (MRH)</td>
<td>A data item holding the latest value encountered in going through a succession of unpredictable values, or simply the latest value obtained as input.</td>
</tr>
<tr>
<td>Most-wanted holder (MWH)</td>
<td>A data item holding the best or otherwise most appropriate value encountered so far.</td>
</tr>
<tr>
<td>Gatherer (GAT)</td>
<td>A data item accumulating the effect of individual values.</td>
</tr>
<tr>
<td>Follower (FOL)</td>
<td>A data item that gets its new value always from the old value of some other data item.</td>
</tr>
<tr>
<td>One-way flag (ONE)</td>
<td>A two-valued data item that cannot revert to its initial value once the value has been changed.</td>
</tr>
<tr>
<td>Temporary (TMP)</td>
<td>A data item holding some value for a very short time only.</td>
</tr>
<tr>
<td>Organizer (ORG)</td>
<td>A data structure storing elements that can be rearranged.</td>
</tr>
<tr>
<td>Container (CNT)</td>
<td>A data structure storing elements that can be added and removed.</td>
</tr>
<tr>
<td>Walker (WLK)</td>
<td>A data item traversing in a data structure.</td>
</tr>
</tbody>
</table>

the behavior of variables with PlanAni has been found to be a powerful way in elaborating students’ role related knowledge [4].

In order to study the emergence of the first general programming plans in a traditionally taught elementary programming course, we conducted a study in fall 2005 in which we followed eight students in their first programming course with program construction protocol tasks [4]. We hypothesized that role plans present basic plan knowledge that novice students gain during an introductory programming course even though it is not taught explicitly. In the study, we also examined the general usability of a role plan analysis model in examining the development of basic programming plans in novice level. On the basis of the results, two major conclusions were made: firstly, the emergence of students’ role plan knowledge was visible in the results, i.e., students learned to use basic programming plans despite these were not explicitly taught. However, their knowledge about the plans was still mostly tacit. Secondly, the role plan analysis model was found as an applicable tool in analyzing the development of plan knowledge. Furthermore, the study revealed an unexpected phenomenon: one role, the stepper was learned considerably better by the students than other roles. We concluded that the explanation for this is that the stepper had a specific name in the course’s vocabulary, “counter”. In other words, this piece of high-level programming knowledge had been explicitly taught in the course and it was indeed the only high level plan the students were conscious of.

To study in detail the development of plan knowledge and the effects of teaching plan knowledge to novices by using roles of variables, we replicated the study in next year’s course. This course was conducted in fall 2006 and in general it was exactly the same as the earlier course in 2005 with the same institute, same instructor and same materials. However, in the 2006 course roles of variables were introduced and PlanAni demonstrations were given to students. Thus, the main objective of this study was to study the emergence of plan knowledge of students who were taught differently in the same course in two consecutive years. The teacher in the course used roles in his teaching for the first time and he was not specifically prepared to teach with roles. In this paper, we report the results of the study.

The rest of the paper is organized as follows. Section 2 describes the study as well as the measures used in the protocol analysis. Section 3 provides the results. In Section 4 the results are discussed followed by the conclusion in Section 5.

2 The Study

In the study, we followed students in their first introductory programming course in imperative programming in fall 2006. Methodologically the study replicates our earlier study conducted in the same course a year earlier [4].

The course used a simplified Pascal-like pseudo lan-
guage that consisted of basic structures of procedural programming languages, with only integer and float type variables and arrays. Only simple programming constructs were taught; e.g., procedures or functions were not included. The course lasted six weeks and consisted of 20 hours of lectures, 18 hours of lab exercises and a two-hour exam. In contrast to the same course a year earlier, roles of variables were gradually introduced in the course and they were used throughout the course in lecture notes, lectures and in exercises. Moreover, PlanAni was used three times in appropriate stages of the course. Otherwise, the course’s schedule and the contents were the same as in the earlier year. The course was conducted outside the researchers’ university in a local polytechnic and neither of the researchers participated in the teaching. The course instructor was an experienced teacher and had been taught the same course (without using roles) for several times before. He was not prepared to teach with roles by the researchers but he was briefly introduced with the role concept and then advised to get introduced to teaching with roles by using the text on teaching roles in [10]. Throughout the course, the course instructor was kept unaware of the exact content of the study sessions as well as the exact details of the research methodology.

The development of plan knowledge of nine students was studied throughout the course by giving them program construction tasks every week during the course; six tasks in total. The first task was given at the second week of the course, which was a simple one, after the basic idea of programming and variables had been taught. At the time of the first task, students had seen the first PlanAni demonstration. In total, students were shown three PlanAni demonstrations. The last task was scheduled at the very end of the course. The sessions were videotaped and the role plan analysis model was applied to the writing protocols.

Participants: In total, nine male students participated in the study. All participants were first-year students in the Degree Programme in Business Information Technology in the North Carelian Polytechnic, Finland. The participants were randomly selected by the course instructor among students having no or little previous background in programming. The participants were given a small monetary compensation after the study.

Materials: For each week a single elementary level verbal programming assignment was prepared. The assignments were exactly the same as in the study conducted in the previous year. The complexity of the problems reflected the prevailing state of the course, thus the first problems were very simple with no loops or complex plans needed while the last tasks were hardest and required several plans in order to reach correct outcome. The phrasing of the tasks was designed carefully so that no procedural information could be extracted from the delineation of the problem. Figure 1 presents the assignment of the third task (translated from Finnish). All six problems had different application domains. The tasks are available in [10].

During the whole study, the optimal solution for two consecutive tasks (task 1 and task 2; task 3 and task 4; task 5 and task 6) was highly similar, i.e., the tasks made up three problem pairs. However, the domains of the two problems in each problem pair were different. Before each problem pair, PlanAni was used in the lab exercises.

In total, six roles were selected for the study. In the first problem pair only two roles, fixed value and gatherer, were needed. Loops were introduced in the third week of the course and from the third problem onwards, implementation of a loop was needed for a correct solution. In the third and fourth tasks two new roles, stepper and most-recent holder, were needed. The last two problems were hardest having the most complex roles needed in the solution; follower and most-wanted holder in addition to the four previous roles.

Procedure: Protocol sessions were conducted with one student at a time. In order to ensure that the students had been given approximately same amount of instruction, the time between sessions was kept as small as possible and the sessions were conducted during the lectures and exercises. Most of the course’s lectures and exercise sessions lasted two hours, thus during a lecture, typically three to four study sessions were accomplished. When a participant returned to the classroom after a session, he was briefed on the issues that he had missed during the session.

Participants were working on a desk in a silent environment and they were provided a pencil, an eraser and several blank sheets of paper for each task. In the beginning of a session, the experimenter first read out the programming assignment and then gave the assignment paper to the participant. The participant was asked to write a program on a paper using the pseudo language introduced in the course. Pencil and paper was used instead of a computer for two reasons; firstly, the course was taught using pseudo language, which was not compilable. Secondly, and more importantly; the uncomfortableness of text editing on a paper should encourage the student to think the correct place of each code line more carefully. Eventually this should reduce the use of trial and error and cause more liable results on students’ true knowledge. The participants were also encouraged to verbalize what they were doing and what

Figure 1. Example programming assignment.
crossed their minds during the program creation, i.e., a talk-
 aloud protocol along with the video capture of the paper was
 collected. If a participant remained silent for several min-
tes, or if a participant clearly stopped to think over some
 issue for a long time, the experimenter prompted to speak
 aloud their thoughts.

 After a participant was finished with the task or was not
 able to continue after a serious try, the experimenter asked
 a few questions concerning participant’s own feelings about
 the task and his performance. If a participant wanted to give
 up with an obviously incomplete solution before the prede-
fined time limit of one hour was reached, he was encouraged
to give a better try.

 Analysis: Protocol analysis was performed by using the
 role plan analysis model introduced in [4] and meant to be
 a tool that can be used in analyzing the amount of program-
 ming knowledge in novice level. The model is based on the
 concept of roles of variables and the theories of program-
ing schemas and plans [1, 2, 5, 6]. Methodologically the
 analysis extends Rist’s theory of schema expansion [6, 7].

 Each role consists of role plan pieces, i.e., program frag-
 ments that have to be written to implement a complete role
 plan. Typically a plan piece corresponds to a single line of
code, e.g., “a=0”. In the role plan analysis model, a pro-
 gramming process is analyzed by examining individually all
 variables with a known role. Results are obtained by ex-
amining the order the plan pieces are written down and by
 looking at discrepancies from the left-to-right writing order
 that is called forward development and known to be used by
 experts in small and not too complex tasks [5].

 Each variable can be scored with three different princi-
 ples. In Strict Forward Development (SFD), which is the
 simplest of the measures and closest to the original idea by
 Rist, implementation of a role plan always represents either
 pure forward development (if all plan pieces are written in
 the correct order), or backward development (if the writing
 order differs from the correct order). Moreover, in novice
 level programs it can be quite reliably defined which role
 plans are needed in order to reach a correct solution. Thus,
 missing role plans are also included in the analysis by defi-
 ning them as backward development. Finally, the measure
 for strict forward development is the proportion of variables
 showing forward development.

 In Partial Forward Development (PFD) the positions of
 the plan pieces in the plan is examined. If the position of
 a plan piece is wrong, the distance between its actual place
 and its correct place decreases the amount of forward de-
 velopment. Thus, small deviations cause only a small decrease
 in the amount of forward development instead of causing
 the whole plan to be treated as backward (as is the case in
 SFD analysis). The final measure is the mean proportion
 of forward development of all variables. Finally, the third
 measure, Pure Partial Forward Development (PPFD), is similar
 to PFD, but missing role plans are not included. The role
 plan analysis model is described in detail in [4].

 The analysis is based on the program writing actions vis-
 ible in the video tapes. The talk-aloud protocols were not
 analyzed systematically—they were mostly used to ensure
 that the student had understood the task correctly.

 3 Results

 Students’ programming processes were analyzed using
 the three different measures of the role plan analysis model
 similarly as in the earlier study in 2005.

 Table 2 shows the proportion of forward development
 for all participants as well as the mean in each task for
 both student groups (NR: students who attended the 2005
 course; R: students who attended the course in fall 2006).
 All three measures (SFD/PFD/PPFD) are shown in the Ta-
 ble. In order to maintain participants’ anonymity, the partic-
 ipant numbers do not reflect the order in which participants
 attended the sessions.

 Strict forward development: In SFD, students who
 were taught with roles (R group) outperformed students
 who were taught with no roles (NR group) in the first five
 tasks. In the last task NR group students were slightly bet-
ter. The difference between the two groups is biggest in the
 first task (NR 8%; R 78%). This difference is statistically
 significant (two-tailed t test, \( t = -6.015, df = 15, p < 0.0001 \)). Differences in other tasks are not statistically sig-
nificant.

 Looking at the three problem pairs, the mean percentages
 for the NR group are 25% in the first, 29% in the second and
 37% in the third problem pair whereas the same numbers
 for the R group are 68%, 40%, and 45%. The difference is
 statistically significant in the first problem pair (two-tailed t
 test, \( t = -3.767, df = 32, p = 0.0007 \)).

 Partial forward development: In the analysis of par-
 tial forward development (PFD) tendencies between the
 two groups follow the results of SFD analysis; the only
 significant difference is in the first task (two-tailed t
 test, \( t = -6.686, df = 15, p < 0.0001 \)). In the pure partial for-
 ward development (PPFD) the proportions of forward de-
 velopment are, expectedly, significantly higher than in SFD
 and PFD. The only significant difference is again in the first
 task (two-tailed t test, \( t = -5.061, df = 15, p = 0.0001 \)). In each problem pair, the mean proportion of forward de-
 velopment is higher with the R group than with the NR group
 in both for PFD and PPFD.

 Role-wise examination: The amount of forward de-
 velopment was also calculated individually for the six roles
 that were selected for the study. Table 3 shows the pro-
 portion of forward development for each role in each task
 as well as the mean percentage for all three measures
 (SFD/PFD/PPFD) in all six tasks for both student groups.
Table 2. Proportion of forward development (SFD/PFD/PPFD) by each participant in each task in the NR group (no roles taught) and in the R group (roles taught).

<table>
<thead>
<tr>
<th>Participant</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
<th>Task 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1-NR</td>
<td>0/0/0</td>
<td>0/46/86</td>
<td>0/28/44</td>
<td>33/67/67</td>
<td>25/32/74</td>
<td>29/72/86</td>
</tr>
<tr>
<td>P2-NR</td>
<td>0/0/0</td>
<td>0/22/48</td>
<td>0/10/50</td>
<td>0/10/50</td>
<td>0/13/50</td>
<td>0/14/71</td>
</tr>
<tr>
<td>P3-NR</td>
<td>0/67/80</td>
<td>33/71/81</td>
<td>36/69/95</td>
<td>25/62/89</td>
<td>33/77/92</td>
<td>63/74/96</td>
</tr>
<tr>
<td>P4-NR</td>
<td>0/51/59</td>
<td>80/94/94</td>
<td>0/67/67</td>
<td>50/90/90</td>
<td>33/88/88</td>
<td>60/98/98</td>
</tr>
<tr>
<td>P5-NR</td>
<td>0/0/0</td>
<td>0/42/49</td>
<td>33/33/100</td>
<td>50/65/94</td>
<td>43/31/94</td>
<td>33/48/100</td>
</tr>
<tr>
<td>P6-NR</td>
<td>67/48/48</td>
<td>100/100/100</td>
<td>0/50/81</td>
<td>25/61/70</td>
<td>75/88/98</td>
<td></td>
</tr>
<tr>
<td>P7-NR</td>
<td>0/0/0</td>
<td>11/50/58</td>
<td>25/25/100</td>
<td>75/73/100</td>
<td>0/0/0</td>
<td>67/83/98</td>
</tr>
<tr>
<td>P8-NR</td>
<td>0/23/83</td>
<td>100/100/100</td>
<td>0/57/77</td>
<td>75/75/100</td>
<td>50/53/91</td>
<td>50/56/100</td>
</tr>
<tr>
<td>Mean (NR)</td>
<td>8/24/34</td>
<td>41/66/77</td>
<td>12/42/75</td>
<td>45/65/83</td>
<td>26/44/69</td>
<td>47/66/93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Participant</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
<th>Task 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1-R</td>
<td>86/88/100</td>
<td>29/62/81</td>
<td>25/14/100</td>
<td>50/43/100</td>
<td>17/24/89</td>
<td>0/8/50</td>
</tr>
<tr>
<td>P2-R</td>
<td>71/83/95</td>
<td>50/67/100</td>
<td>25/46/67</td>
<td>50/92/92</td>
<td>45/64/79</td>
<td>50/69/89</td>
</tr>
<tr>
<td>P3-R</td>
<td>100/100/100</td>
<td>50/84/84</td>
<td>0/71/71</td>
<td>75/97/97</td>
<td>57/95/95</td>
<td>57/71/98</td>
</tr>
<tr>
<td>P4-R</td>
<td>17/85/85</td>
<td>100/100/100</td>
<td>0/21/63</td>
<td>57/71/85</td>
<td>86/86/98</td>
<td>43/77/77</td>
</tr>
<tr>
<td>P5-R</td>
<td>86/85/100</td>
<td>20/24/100</td>
<td>33/27/100</td>
<td>40/83/83</td>
<td>29/70/81</td>
<td>50/76/90</td>
</tr>
<tr>
<td>P6-R</td>
<td>86/85/100</td>
<td>80/93/93</td>
<td>40/65/65</td>
<td>83/98/98</td>
<td>57/90/90</td>
<td>50/94/94</td>
</tr>
<tr>
<td>P7-R</td>
<td>86/85/100</td>
<td>80/78/78</td>
<td>67/85/85</td>
<td>60/74/74</td>
<td>100/100/100</td>
<td>83/94/94</td>
</tr>
<tr>
<td>P8-R</td>
<td>86/85/100</td>
<td>25/92/92</td>
<td>33/27/100</td>
<td>25/67/67</td>
<td>30/51/80</td>
<td>29/44/89</td>
</tr>
<tr>
<td>P9-R</td>
<td>86/85/100</td>
<td>83/83/100</td>
<td>0/0/0</td>
<td>50/65/94</td>
<td>0/13/100</td>
<td>17/13/100</td>
</tr>
<tr>
<td>Mean (R)</td>
<td>78/87/98</td>
<td>57/76/92</td>
<td>25/40/72</td>
<td>54/77/88</td>
<td>47/66/90</td>
<td>42/61/87</td>
</tr>
</tbody>
</table>

In the R group, fixed value has the largest proportion of forward development in all three measures (SFD 79%, PFD 88%, PPFD 93%). This clearly contradicts with the results of the NR group where the stepper has the largest SFD value (50%) and fixed value comes third after most-recent holder (FIX SFD 38%). The hardest role in both groups was follower (FOL SFD in the NR group 7%; in the R group 11%).

The clearest differences in role-wise examination between the groups consider fixed value and most-wanted holder. In SFD analysis, the mean proportion of MWH:s implemented forwardly is 43% with the R group whereas the same amount with the NR group is only 13%. In PFD analysis, which is the most sensitive of the three measures, the difference between the MWH:s implemented by the NR group and by the R group is also distinctive (MRH PFD in the NR group 53%; in the R group 75%).

When looking at the proportion of PFD with most-wanted holders individually, i.e., the mean percentages of PFD for each participant, the mean proportion of MWH:s implemented by the NR group students is 41.6% while with the R group students the same value is 72.8%. This difference is statistically significant (two-tailed t test, \( t = -2.205, df = 32, p = 0.0348 \)).

4 Discussion

In this study, we examined the development of programming plan knowledge of nine students in a programming course where the roles of variables concept was used in teaching (the R group). We also compared the results with an earlier study, which was conducted within the same course in previous year but with no special treatment of roles in teaching (the NR group). We will now discuss the results first from a general point of view and then look at individual roles.

4.1 General Examination

The most interesting result in this study is obviously not the overall performance of the students itself, but the differences between students who have been taught by the same teacher but with a different method. In overall, the performance of the R group students was slightly better than the performance of the NR group students; when looking at the mean proportion of forward development in the problem pairs, the R group students outperformed the NR group students in all measures. However, none of the overall differences is statistically significant. Still, the NR group reached the level of the R group only in the third and in the sixth
Table 3. Proportion of forward development (SFD/PFD/PPFD) of the role plans in overall for the NR group (no roles taught) and for the R group (roles taught).

<table>
<thead>
<tr>
<th>Role</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
<th>Task 6</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAT-NR</td>
<td>0/14/48</td>
<td>38/62/100</td>
<td>0/25/67</td>
<td>38/67/91</td>
<td>13/42/89</td>
<td>38/63/96</td>
<td>21/46/82</td>
</tr>
<tr>
<td>MRH-NR</td>
<td>-</td>
<td>-</td>
<td>13/59/76</td>
<td>63/82/82</td>
<td>33/46/66</td>
<td>67/93/93</td>
<td>44/70/79</td>
</tr>
<tr>
<td>STP-NR</td>
<td>-</td>
<td>-</td>
<td>0/9/46</td>
<td>63/83/96</td>
<td>50/68/91</td>
<td>88/84/98</td>
<td>50/61/83</td>
</tr>
<tr>
<td>MWH-NR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0/34/78</td>
<td>25/72/94</td>
<td>13/53/86</td>
</tr>
<tr>
<td>FOL-NR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0/8/83</td>
<td>13/13/100</td>
<td>7/11/92</td>
</tr>
</tbody>
</table>

| Mean (NR) | 5/22/57 | 38/66/90 | 14/46/71 | 44/64/84 | 29/47/84 | 48/68/97 | 29/51/84 |

| Problem pair mean (NR) | 22/44/73 | 29/55/77 | 38/58/90 |

| FIX-R   | 88/98/98  | 68/85/94  | 100/100/100 | 57/67/82 | 70/81/90 | 89/95/95 | 79/88/93 |
| GAT-R   | 22/46/100 | 25/59/81  | 0/15/50     | 20/68/86 | 30/65/93 | 10/39/78 | 18/49/81 |
| MRH-R   | -         | -         | 33/58/75    | 53/75/81 | 73/80/96 | 42/71/86 | 50/71/85 |
| STP-R   | -         | -         | 20/34/72    | 50/78/88 | 36/72/87 | 56/71/92 | 41/64/85 |
| MWH-R   | -         | -         | -           | -        | 36/76/83 | 50/74/91 | 43/75/87 |
| FOL-R   | -         | -         | -           | -        | 11/24/80 | 11/26/79 | 11/25/80 |

| Mean (R) | 55/72/99 | 47/72/88  | 38/52/74    | 45/72/84 | 43/66/88 | 43/63/87 | 40/62/85 |

| Problem pair mean (R) | 51/72/93 | 42/62/79  | 43/65/88 |

task and although in the last task the overall performance of the NR group was slightly better, the difference between the groups was minor. Then again, the R group outperformed the NR group in all other tasks and some of the differences were significant. It thus seems, that the roles of variables concept had indeed a positive effect in learning programming, even though the overall effect is not statistically significant.

At first glance it seems that the R group students would have had more previous programming knowledge in the beginning of the course than the NR group students—the difference in the first task is evident. However, by the time of the first task, six hours of lectures had already been given, which should be sufficient for both groups to learn the appropriate amount of programming needed to accomplish the first, very easy task. Also, if the R group students would have had previous programming knowledge they should have performed better at least in the tasks two and three, which were still quite simple. However, in those tasks the difference between the two groups was considerably smaller. Furthermore, there were indications of some previous programming knowledge among the NR group students also—some of the students in this group implemented constructs that were not included in the course.

We therefore think that the most plausible explanation for the superior performance of the R group students in the first task is that the most simple demonstration program in PlanAni is practically identical to the solution needed in the first protocol task. The first PlanAni use occurred right before the first protocol task and, thus, participants were probably able to recall the demonstration program still in the protocol sessions. Thus, we argue that although we did not perform accurate testing of the level of students' prior programming knowledge, there is no evidence of major differences in the initial level of knowledge between the NR group participants and the R group participants. The difference in the first task is mainly due to the use of PlanAni.

The effect of PlanAni demonstrations should not, however, be considered as a methodological flaw of the study. It is very unlikely that students would have memorized the program in the first animation. Thus the indisputably better performance of the R group must indicate true learning in programming knowledge. This is supported by the fifth task, where the difference between the two groups is also clear and which was also preceded by a PlanAni demonstration. In this case the PlanAni demonstration program was, however, not related to the protocol task solution and hence the effect cannot be explained by rote learning. Rather, the effect is caused by a combination of the introduction of role knowledge and its elaboration by PlanAni. Moreover, in this phase of learning, PlanAni does not present any new concepts but only deepens the role knowledge.

When looking at the mean proportion of forward development in the three problem pairs, there is an upward trend with the NR group in all measures. In the R group the proportion of forward development is highest in the first problem pair and lowest in the second problem pair until it rises again when moving onwards to the third problem pair. This means that in both groups eventually the performance of students gets better when moving onwards—the overall per-
formance rises even though the tasks get more complex at the same time. The high performance of the R group in the first problem pair, which biases the overall result, is caused by the strong learning effect described above.

When examining the overall use of variables, it seems that the students in the R group had a more data driven approach to programming than students in the NR group. For example, in most cases the students in the R group first declared all variables they knew they needed and only after that began to write other code lines. Moreover, the poorest performers in the R group were usually able to declare most variables they would need and name them so that the purpose of each variable was clear although in some cases they could not write any appropriate code lines. In the NR group, however, the poorest performers came up with a blank paper only, or provided sketches without any trace of program code.

4.2 Role-wise Examination

Roles of variables are cognitive concepts and different people may see the roles differently, thus a reliable comparison of the mental complexity between roles is hard and knowledge about the complexity of roles is based on empirical evidence. However, some expectations about the theoretical complexity of roles may be done. For example, the behavior of fixed value is most certainly the easiest to understand and it is typically used first in any programming course. Then again, most-recent holder should be easier than stepper and gatherer because there is no calculation or compulsory initialization needed. Furthermore, studies with novices have shown that most-wanted holder and follower are the hardest behaviors to learn [3, 4]. The results of the role-wise SFD examination show that the order of the roles is much closer to these expectations with the R group than with the NR group. Moreover, students in the R group performed better with all roles, also with the hardest ones.

There is a clear difference between the two groups in the use of fixed value. It was learned fast by the R group and the performance was high already in the first task (SFD 88%) whereas students in the NR group did not learn to use the role during the first tasks so well. In overall, fixed value was—expectedly—the easiest role for the R group students (SFD 79%), but in the NR group fixed value was in the third place (SFD 38%). Fixed value is the simplest of roles and understanding it correlates more or less with understanding the concept of a variable in general. Therefore, it seems that students in the R group learned the basic notion of variables faster than students in the NR group although students in the R group did not yet have knowledge about more complex plans, e.g., gatherer, which would have been needed in the first task, also.

The distribution of SFD between roles within the NR group did not follow the expected behavior and the stepper had the highest value (50%) of all roles. Stepper was, however, in a special position for the NR group, because it was the only high-level plan that had a name in the course’s vocabulary and was, therefore, the only plan that was explicitly taught. Thus, the role of a stepper was emphasized in the course and students’ performance with steppers was higher than otherwise expected. In the R group, however, stepper was taught along with other role plans and had, thus, no special treatment when compared to other roles. In the R group stepper was in the fourth place, which corresponds its complexity among other roles.

The proportion of forward development with gatherers was unexpectedly low for both groups although it was needed in all tasks. In the first two tasks only a few students implemented a gatherer, because a correct outcome could be reached also by using just fixed values—however, this approach is much more laborious. The low performance with gatherers continued through the whole course with both groups. Moreover, the big differences between PFD and PPFD values indicate that the reason for the low performance was indeed the complete lack of gatherers. Theoretically gatherer should not be such a difficult plan—therefore this phenomenon still remains without a proper explanation.

The most-wanted holder is one of the hardest roles for novice students to implement forwardly. The most usual mistakes are in the initialization and in the comparison of the new value with the current best value. The difference between the groups with most-wanted holder was statistically significant; the R group overperformed the NR group. This suggests that because the students in the R group learned to use the easiest roles quickly in the beginning of the course, they had the capacity to learn the hardest roles better. While students in the NR group still struggled with the easier plans in the last two tasks, students in the R group were able to learn new plans and succeeded in implementing them more often.

The follower was expectedly the hardest role for both groups and most students did not learn to use the role at all. However, although there is only a minor difference in SFD values of followers between the two groups, the PFD value for followers in the R group is higher than in the NR group (PFD-NR 11%, PFD-R 25%).

5 Conclusions

We have presented a study where the learning outcome of a programming course where roles of variables were used as a pedagogical method was compared with the outcome of a course taught traditionally by the same teacher. Development of programming knowledge was measured with the role plan analysis model that is an extension and refinement of Rist’s theory of schema expansion. Although the
number of participants was small, the overall results show that students who were taught with roles learned the basic notion of variables faster—from the very beginning they used fixed value plan considerably better than students who were taught only with traditional methods. Students taught with roles were also more able to understand and implement harder roles at later phases of the programming course.

It seems that the traditional teaching gave nearly the same final learning result than teaching with roles, but roles gave the students the needed knowledge earlier. Moreover, students who were introduced to roles were able to implement more correct plans during the whole course. It must be remembered that the course was relatively long with only the simplest programming constructs taught using a simplified language, so there was no need to learn details of notation or exact syntax of a programming language. Thus, it is only natural that students did finally learn to write simple programs well enough with traditional teaching methods, also. This explains the fact that the difference between the two groups got smaller as the course proceeded.

This study has confirmed earlier findings on the positive effects of the use of the role concept in elementary programming education—this time with a teacher not belonging to the research group and having no previous knowledge of the roles. The current study has moreover shed light on differences in the development of role plan knowledge when roles are or are not explicitly used in teaching. The study focused on the very first programming course and it would be interesting to study long-term effects of roles on later programming skill development when programming tasks become more complex and require more skilled programming.

6 Acknowledgements

Sincere thanks to Mr. Eero Mönkkönen for the remarkable positive responsiveness and open-mindedness towards our research and for giving us the opportunity to conduct the studies in two consecutive years. Thanks also to North Carelian Polytechnic for the permission to perform the studies.

This work was supported by the Academy of Finland under grant number 206574.

References