A Role-Based Analysis Model for the Evaluation of Novices' Programming Knowledge Development

Pauli Byckling and Jorma Sajaniemi

Outline

- The roles of variables concept
- Rist's basic plan development
- The new model—role plan development
- Validation of the model—the study
- Results
- Conclusion

JOENSUUN YLIOPISTO
University of Joensuu

Representations of High Level Programming Knowledge

- Schemas and Programming plans (Adelson 1981,1984; Soloway et al. 1982; Rist 1989; etc.)
  - Schema: a *data structure* which represents generic concepts stored in human memory (Détienne 2002)
  - Programming plan: a sequence of *actions* in a program which will achieve the *goal* of the program (Détienne 2002)

- Roles of Variables (Sajaniemi 2002)
  - Stereotypical behavior patterns of variables and attributes; occur in programs over and over again
  - A way to represent high level *programming knowledge*

Roles of Variables 1/2

- Roles of Variables (Sajaniemi 2002)
  - A way to represent high level *programming knowledge*
  - Stereotypical behavior patterns of variables and attributes; occur in programs over and over again

- Examples:

Example 1 — Counter:	Example 2 — Running total:

`i = 0;`
`. . .`
`i++;`
→ Generalized to *stepper*
	`rainsum = 0;`
	`. . .`
	`rainsum = rainsum + rain;`
	→ Generalized to *gatherer*

- Roles are a part of experts' tacit knowledge (Sajaniemi and Navarro Prieto 2005)

- Explicit teaching of roles resulted in better programming skills (Byckling and Sajaniemi 2006) and in better mental models of programs (Sajaniemi and Kuittinen 2005).

Roles of Variables 2/2

- Revised set of roles; applicable in object-oriented, procedural and functional programming

Role	Example	Coverage in novice-level programs
Fixed value	`maxStringLength`	99%
Stepper	`count`	
Most-recent holder	`inputData`	
Most-wanted holder	`maximum`	
Gatherer	`sum`	
Follower	`prev`	
One-way flag	`errorsOccurred`	
Temporary	`temp`	
Organizer	`sortArray`	
Container	`processQueue`	
Walker	`currNode`	
Other	`tabPos`	1%

Rist's Basic Plan Development

- Theory of schema expansion (Rist, 1989)
- Programming consists of implementation of plans (five basic plans)
- Plans consist of plan pieces: *Initialization, Calculation, Output*
- Analysis of the writing order of plan pieces
  - Forward development (schema expansion): plan pieces retrieved from memory, writing order reflects the final form of the program code
  - Backward development (focal expansion) : plan pieces created during programming, writing order goes from calculation to initialization

Our New Model—Role Plan Development

- Methodology and the basic idea of plan development based on Rist's work
- Roles as programming plans that represent knowledge about variables
- Analysis of the writing order of *role plan pieces*: *Goal, Declaration, Initialization, Extension, Computation, Use of the latest value, Use of the final value*
- Examination of each plan focuses only on lines directly related to the variable in question
- Advantages:
  - More detailed with more plan pieces
  - Based on roles of variables → more systematic programming knowledge coverage

Actions in the Role Plan Analysis Model

- Actions that execute the role plan pieces defined individually for all roles
- Example 1, Stepper:
  - (D)eclaration: variable declaration (if needed)
  - (I)nitialization: initialization with initial value
  - (E)xtension: possible use in controlling a loop
  - (C)omputation: update in the loop
  - (UC) Use of the current value: use in the loop
  - (UF) Use of the final value: use after the loop, if any
- Example 2, One-way flag:
  - (D)eclaration: variable declaration (if needed)
  - (I)nitialization: initialization (typically with 'false' or 0)
  - (E)xtension: check of the condition which affects the flag
  - (C)omputation: update if the condition is met
  - (UC) Use of the current value: use of the flag (e.g. in controlling a loop), if any
  - (UF) Use of the final value: use of the final state of the flag

Role Plan Analysis Model—Scoring

- Scoring, basic idea
  - Theoretical order (G-D-I-E-C-UC-UF): Forward development
  - Order of the lines in final program: Forward development
  - Any other order: Backward development
  - “No development” (ND)

- Example:

Line number	Program code	Piece type
3.	`var hours, minutes: integer;`	Declaration
9.	`readln(hours);`	Computation
8.	`while(hours > 0) or (minutes > 0 do)`	Extension
11.	`transformation := 60 * hours + minutes;`	Use of the latest value

→ backward development

Role Plan Analysis Model—More Scoring

- Strict forward development (SFD)
  - previous slide
- Partial forward development (PFD)
  - sequence number for each compulsory plan piece → appearance order
  - jumps between pieces
    (backward = 0, forward = 1/length ot the jump)
  - result: amount of forward development
- Pure partial forward development (PPFD)
  - similar to PFD, but ND's are not taken into account
- Scoring example:

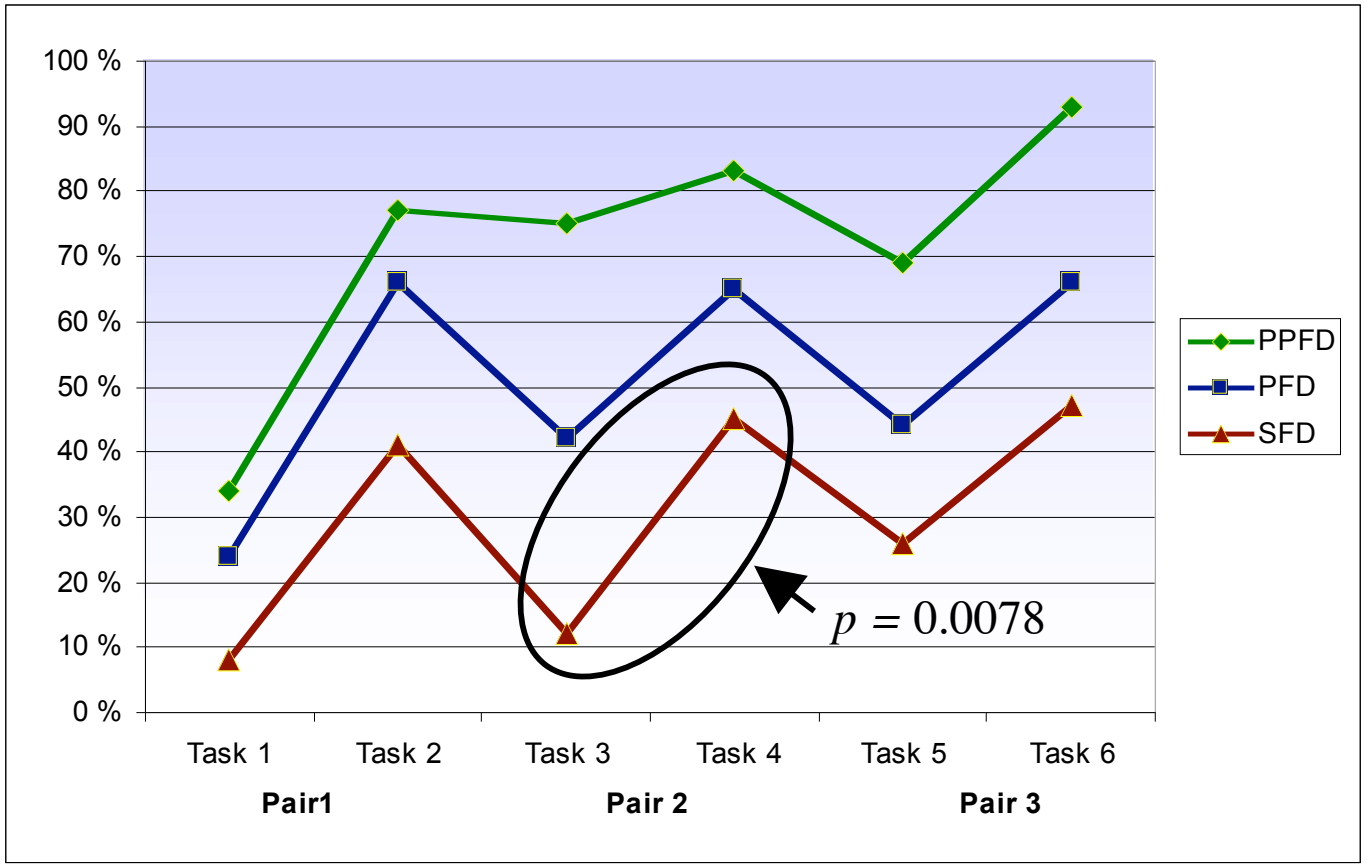
	Sequence	**SFD**	**PFD**	**PPFD**
Var1	(0),1,2,3,4,5	**1**	1, 1, 1, 1, 1 = **5/5**	**5/5**
Var2	(0),3,1,2,4,5	**0**	0.33, 0, 1, 0.5, 1 = **2.83/5**	**2.83/5**
Var3	Missing	**0**	**0/5**	**-**
Score		**1/3 = 33 %**	**7.83/15 = 52%**	**7.83/10 = 78 %**

Validation of the Model—the Study

- Eight students in their first introductory programming course (in North Carelian Polytechnic, 2005)
  - Imperative programming with Pascal-like pseudo language
  - No roles or any other plans taught
- Weekly program creation protocol tasks, six tasks in total (three problem pairs)
- Analysis with the role plan analysis model

Results 1/2

- Overall results

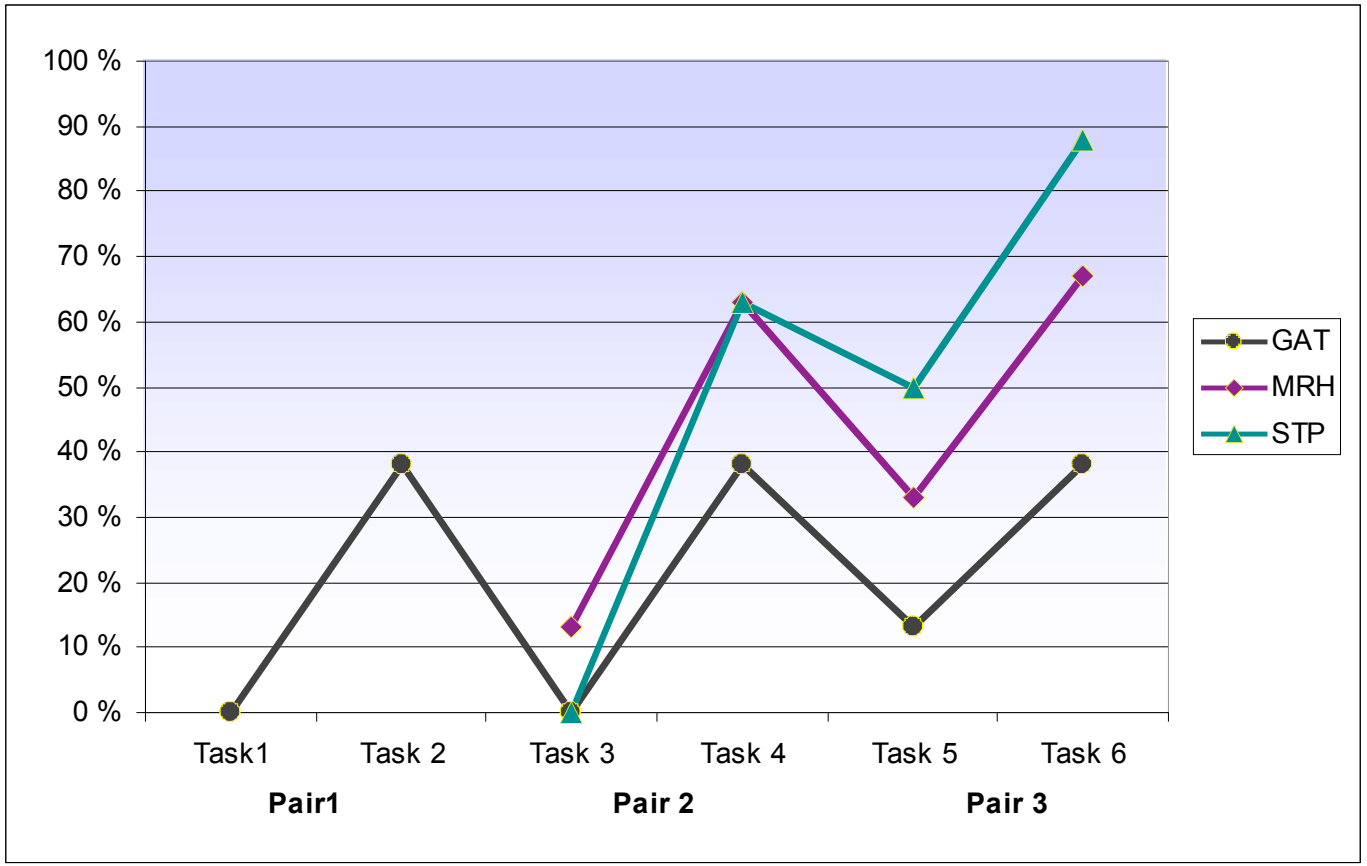


JOENSUUN YLIOPISTO
University of Joensuu

University of Joensuu
www.joensuu.fi

Results 2/2

- SFD, STP vs. MRH/GAT:



Discussion

- Clear upward trend in overall
  - biggest differences between tasks within problem pairs
- The results reflect the expected behavior
- High amount of correct steppers
  - reflects the explicit teaching of the counter plan?
- Different versions of the model:

a large value in PPFD →	ability to use familiar plans
a large value in PFD →	ability to learn new plans
a large value in SFD →	ability to apply both familiar and new plans effectively

Conclusion

- Emergence of students' role plan knowledge is visible in the results

  → the analysis model is applicable in analyzing the development of plan knowledge

- Future work: classroom experiment in fall 2006:
  - introduction of roles of variables in the course
  - similar protocol tasks and analysis

References

- B. Adelson. Problem solving and the development of abstract categories in programming languages. Memory and Cognition, 9(4):422–433, 1981.
- B. Adelson. When novices surpass experts: The difficulty of a task may increase with expertise. Journal of Experimental Psychology: Learning, Memory and Cognition, 10(3):483–495, 1984.
- P. Byckling, J. Sajaniemi. Roles of Variables and Programming Skills Improvement. Proceedings of the 37th SIGCSE Tehnical Symposium on Computer Science Education (SIGCSE 2006), Association for Computing Machinery, 413-417, 2006.
- F. Détienne. Program understanding and knowledge organization: the influence of acquired schemas. In P. Falzon, editor, Cognitive Ergonomics: Understanding, Learning and Designing Human-Computer Interaction, pages 245–256. Academic Press, 1990.
- J. Sajaniemi. An empirical analysis of roles of variables in novice-level procedural programs. In Proceedings of IEEE 2002 Symposia on Human Centric Computing Languages and Environments (HCC'02), pages 37–39. IEEE Computer Society, 2002.
- J. Sajaniemi and M. Kuittinen. An experiment on using roles of variables in teaching introductory programming. Computer Science Education, 15(1):59–82, 2005.
- J. Sajaniemi and R. Navarro Prieto. Roles of variables in experts' programming knowledge. In P. Romero, J. Good, S. Bryant, and E. A. Chaparro, editors, Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group (PPIG 2005), pages 145–159, 2005.
- E. Soloway, K. Ehrlich, J. Bonar, and J. Greenspan. What do novices know about programming? In A. Badre and B. Shneiderman, editors, Directions in Human Computer Interaction. Ablex Publishing Corporation, 1982.
- R. S. Rist. Schema creation in programming. Cognitive Science, 13:389–414, 1989.