comprehension has increased our understanding of the comprehension processes and, moreover, of the mental representations of imperative programs and imperative programming knowledge. The structure of OO programs differs so much from imperative and procedural programs that one may presume that their comprehension processes do also differ considerably. Again, some elements (e.g., hypothesis-driven comprehension) are the same, but issues related to program structure can be assumed to differ. We therefore suggest research into *experts' and novices' OO program comprehension processes.*

Finally, results of the research suggested above and summarized in Table 3 should be utilized in devising effective methods for teaching OOP. However, we do not include this work in the research agenda proposal for two reasons. Firstly, the right time for such educational-oriented research will come only after there is a large body of results obtained from the research agenda. Secondly, it may well be that effective ways to transfer experts' mental representations, skills and strategies are at least partially revealed during the earlier research covered by the agenda.

Table 3. Proposal for Research Agenda in OOP and OOP Education.

	Performance			Development	
Topic of investigation	**Expert**	**Expert vs. Novice**	**Novice**	**Cognition**	**Education**
Mental representation of notional machine	•	•		•	•
Mental representation of OO programs	•	•		•	
Mental representation of OOP		•		•	
Program comprehension	•	•	•	•	
Tracing and debugging		•		•	
Program design		•		•	

CONCLUSION

In programming education, there has been a major shift in the programming paradigm used in the first courses. To please industry and students, educators have moved from imperative and procedural programming to object-orientation without studying its necessity or consequences and without studying how OOP education should be carried out. Moreover, classic results from imperative and procedural programming have been used as such even though their applicability in the OO case can be questioned. The shift from imperative/procedural

programming to object-orientation is so revolutionary that the use of research results obtained in the imperative and procedural cases is doubtful in the OO case. The number of notations and concepts needed, the size of the notional machine required, and the whole orientation of programming are so different that the basic assumptions used in imperative and procedural programming research do not necessarily hold for object-orientation. Even though some results may apply in object-orientation, there is a need to find out on what occasions this happens to be the case.

There is a lack of systematic research into the fundamental cognitive and educational issues in learning and teaching OOP. Lister et al. (2006, p. 160) conclude their paper by noting that "our community needs to discuss—and debate—this issue," but we claim that the computer science education research community and the psychology of programming community need to rigorously study these issues first. For that purpose, we have presented a research agenda comprising

- *Constructing a model of the OOP expert:* experts' mental representations of the notional OO machine; exploratory research into experts' mental representations of OO programs
- *Understanding the differences between OOP experts and novices:* experts' and novices' differences in mental representations, program comprehension processes, skills and strategies within OOP
- *Fostering OOP novices' cognitive development:* novices' cognitive development in OOP; ways to convey the notional OO machine to novices

High dropout rates from OOP courses and poor learning outcomes pose problems to students, educators, and educational institutions. These problems can be attacked only with rigorous research into the psychological and educational issues involved.

ENDNOTES

1. Imperative and procedural programming are often considered synonyms, but in this paper *imperative* refers to programming with variables, assignment, and simple imperative control structures, such as sequence, iteration, and conditionals, whereas *procedural* covers procedures, parameters and recursion, also.
2. Here we are interested in differences that are inherent to object-orientation and the way object-related concepts are implemented in Java. We do not treat Java problems that occur within the imperative parts of Java, for example, that using "=" as the assignment operator makes some students to confuse assignment with mathematical equality.
3. In this literature review, we look at programming only. Thus, we do not include system design literature even though we do include program design literature.

REFERENCES

Anderson, J. R. (2000). *Cognitive psychology and its implication*s (5th ed.). New York: Worth Publishers.

Bednarik, R., & Tukiainen, M. (2007). Analysing and interpreting quantitative eye-tracking data in the studies of programming: Phases of debugging with multiple representations. In J. Sajaniemi, M. Tukiainen, R. Bednarik, & S. Nevalainen (Eds.), *Proceedings of the 19th Annual Workshop of the Psychology of*

Programming Interest Group (pp. 158–172). Joensuu, Finland: University of Joensuu, Department of Computer Science and Statistics.

Ben-David Kolikant, Y., & Haberman, B. (2001). Activating "black boxes" instead of opening "zippers": A method of teaching novices. In *ITiCSE '01: Proceedings of the Sixth Annual Conference on Innovation and Technology in Computer Science Education* (pp. 41–44). New York: ACM Press.

Bennedsen, J., & Caspersen, M. E. (2004). Programming in context: A model-first approach to CS1. In *SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (pp. 477–481). New York: ACM Press.

Bierre, K., Ventura, P., Phelps, A., & Egert, C. (2006). Motivating OOP by blowing things up: An exercise in cooperation and competition in an introductory Java programming course. In *SIGCSE '06: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (pp. 354–358). New York: ACM Press.

Brooks, R. E. (1983). Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies*, *18*, 534–554.

Cassel, L. B., McGettrick, A., Guzdial, M., & Roberts, E. (2007). The current crisis in computing: What are the real issues? In *SIGCSE '07: Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (pp. 329–330). New York: ACM Press.

Chen, T.-Y., Monge, A., & Simon, B. (2006). Relationship of early programming language to novice generated design. In *SIGCSE '06: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (pp. 495–499). New York: ACM Press.

Cooper, D., & Clancy, M. (1982). *Oh! Pascal!* New York: W. W. Norton & Company.

Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. In *SIGCSE '03: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education* (pp. 191–195). New York: ACM Press.

Corritore, C. L., & Wiedenbeck, S. (1991). What do novices learn during program comprehension? *International Journal of Human-Computer Interaction*, *3*, 199–222.

Corritore, C. L., & Wiedenbeck, S. (1999). Mental representations of expert procedural and object-oriented programmers in a software maintenance task. *International Journal of Human-Computer Studies*, *50*, 61–83.

Davies, S. P. (1993). Models and theories of programming strategy. *International Journal of Man-Machine Studies*, *39*, 237–267.

Davies, S. P., Gilmore, D. J., & Green, T. R. G. (1995). Are objects that important? Effects of expertise and familiarity on classification of object-oriented code. *Human-Computer Interaction*, *10*, 227–248.

Détienne, F. (1997). Assessing the cognitive consequences of the object-oriented approach: A survey of empirical research on object-oriented design by individuals and teams. *Interacting with Computers*, *9*, 47–72.

Détienne, F. (2002). *Software design: Cognitive aspects*. London: Springer-Verlag.

de Raadt, M., Watson, R., & Toleman, M. (2002). Language trends in introductory programming courses. In E. Cohen & E. Boyd (Eds.), *Proceedings of Informing Science and IT Education Conference* (pp. 329–337). Santa Rosa, CA, USA: Informing Science Institute.

du Boulay, B. (1989). Some difficulties of learning to program. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 283–299). Hillsdale, NJ, USA: Lawrence Erlbaum Associates.

du Boulay, B., O'Shea, T., & Monk, J. (1981). The black box inside the glass box: Presenting computing concepts to novices. *International Journal of Man-Machine Studies*, *14*, 237–249.

Ebrahimi, A., & Schweikert, C. (2006). Empirical study of novice programming with plans and objects. *SIGCSE Bulletin*, *38*(4), 52–54.

Eckerdal, A., & Thuné, M. (2005). Novice Java programmers' conceptions of "object" and "class", and variation theory. In *ITiCSE '05*: *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 89–93). New York: ACM Press.

Fleury, A. E. (1991). Parameter passing: The rules the students construct. *SIGCSE Bulletin, 23*(1), 283–286.

Fleury, A. E. (2000). Programming in Java: Student-constructed rules. In *SIGCSE '00: Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education* (pp. 197–201). New York: ACM Press.

Gilmore, D. J., & Green, T. R. G. (1984). Comprehension and recall of miniature programs. *International Journal of Man-Machine Studies*, *21*, 31–48.

Green, T. R. G., Bellamy, R. K. E., & Parker, J. M. (1987). Parsing and gnisrap: A model of device use. In G. M. Olson, S. Sheppard, & E. Soloway (Eds.), *Empirical studies of programmers: Second workshop* (pp. 132–146). Norwood, NJ, USA: Ablex Publishing Company.

Gries, P., & Gries, D. (2002). Frames and folders: A teachable memory model for Java. *The Journal of Computing Sciences in Colleges*, *17*(6), 182–196.

Holland, S., Griffiths, R., & Woodman, M. (1997). Avoiding object misconceptions. *SIGCSE Bulletin*, *29*(1), 131–134.

Holliday, M. A., & Luginbuhl, D. (2004). CS1 assessment using memory diagrams. In *SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (pp. 200–204). New York: ACM Press.

Hsia, J. I., Simpson, E., Smith, D., & Cartwright, R. (2005). Taming Java for the classroom. In *SIGCSE '05: Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (pp. 327–331). New York: ACM Press.

Kinnunen, P., & Malmi, L. (2006). Why students drop out CS1 course? In *ICER '06: Proceedings of the 2006 International Workshop on Computing Education Research* (pp. 97–108). New York: ACM Press.

Kölling, M., & Henriksen, P. (2005). Game programming in introductory courses with direct state manipulation. In *ITiCSE '05: Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 59–63). New York: ACM Press.

Lee, A., & Pennington, N. (1994). The effects of programming on cognitive activities in design. *International Journal of Human-Computer Studies*, *40*, 577–601.

Letovsky, S. (1986). Cognitive processes in program comprehension. In E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmers* (pp. 58–79). Norwood, NJ: Ablex Publishing Company.

Levy, D. (2001). Insights and conflicts in discussing recursion: A case study. *Computer Science Education*, *11*, 305–322.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B., & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bulletin*, *36*(4), 119–150.

Lister, R., Berglund, A., Clear, T., Bergin, J., Garvin-Doxas, K., Hanks, B., Hitchner, L., Luxton-Reilly, A., Sanders, K., Schulte, C., & Whalley, J. L. (2006). Research perspectives on the objects-early debate. *SIGCSE Bulletin*, *38*(4), 146–165.

Lopez-Herrejon, R. E., & Schulman, M. (2004). Using interactive technology in a short Java course: An experience report. In *ITiCSE '04: Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 203–207). New York: ACM Press.

Mahmoud, Q. H., Dobosiewicz, W., & Swayne, D. (2004). Redesigning introductory computer programming with HTML, JavaScript, and Java. In *SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (pp. 120–124). New York: ACM Press.

Marrero, W., & Settle, A. (2005). Testing first: Emphasizing testing in early programming courses. In *ITiCSE '05: Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 4–8). New York: ACM Press.

McCracken, M., Wilusz, T., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Ben-David Kolikant, Y., Laxer, C., Thomas, L., & Utting, I. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *SIGCSE Bulletin, 33*(4), 125–140.

Mead, J., Gray, S., Hamer, J., James, R., Sorva, J., Clair, C. S., & Thomas, L. (2006). A cognitive approach to identifying measurable milestones for programming skill acquisition. *SIGCSE Bulletin*, *38*(4), 182–194.

Moreno, A., Myller, N., Sutinen, E., & Ben-Ari, M. (2004). Visualizing programs with Jeliot 3. In *AVI '04: Proceedings of the Working Conference on Advanced Visual Interfaces* (pp. 373–376). New York: ACM.

Pennington, N. (1987). Comprehension strategies in programming. In G. M. Olson, S. Sheppard, & E. Soloway (Eds.), *Empirical studies of programmers: Second workshop* (pp. 100–113). Norwood, NJ, USA: Ablex Publishing Company.

Pennington, N., Lee, A., & Rehder, B. (1995). Cognitive activities and levels of abstraction in procedural and object-oriented design. *Human-Computer Interaction*, *10*, 171–226.

Perkins, D. N., & Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. In E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmers* (pp. 213–229). Norwood, NJ, USA: Ablex Publishing Company.

Radenski, A. (2006). "Python first": A lab-based digital introduction to computer science. In *ITICSE '06: Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 197–201). New York: ACM Press.

Ragonis, N., & Ben-Ari, M. (2005). A long-term investigation of the comprehension of OOP concepts by novices. *Computer Science Education*, *15*, 203–221.

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

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*, 137–172.

Romero, P., Lutz, R., Cox, R., & du Boulay, B. (2002). Co-ordination of multiple external representations during Java program debugging. In *Proceedings of the IEEE 2002 Symposia on Human Centric Computing Languages and Environments* (pp. 207–214). Los Alamitos, CA, USA: IEEE Computer Society Press.

Rosson, M. B., & Gold, E. (1989). *Problem-solution mapping in object-oriented design*. New York: IBM T. J. Watson Research Center.

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.

Sajaniemi, J., Byckling, P., & Gerdt, P. (2006). Metaphor-based animation of OO programs. In *SoftVis '06: Proceedings of the ACM Symposium on Software Visualization* (pp. 173–174). New York: ACM Press.

Sajaniemi, J., Kuittinen, M., & Tikansalo, T. (2007). A study of the development of students' visualizations of program state during an elementary object-oriented programming course. In *ICER '07: Proceedings of the Third International Workshop on Computing Education Research* (pp. 1–15). New York: ACM Press.

Samurçay, R. (1989). The concept of variable in programming: Its meaning and use in problem-solving by novice programmers. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 161–178). Hillsdale, NJ, USA: Lawrence Erlbaum Associates.

Shanmugasundaram, V., Juell, P., & Hill, C. (2006). Knowledge building using visualizations. In *ITICSE '06: Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 23–27). New York: ACM Press.

Sharp, H., & Griffyth, J. (1999). The effect of previous software development experience on understanding the object-oriented paradigm. *Journal of Computers in Mathematics and Science Teaching*, *18*, 245–265.

Soloway, E., & Spohrer, J. C. (Eds.). (1989). *Studying the novice programmer*. Hillsdale, NJ, USA: Lawrence Erlbaum Associates.

Spohrer, J. C., Soloway, E., & Pope, E. (1989). A goal/plan analysis of buggy Pascal programs. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 355–399). Hillsdale, NJ, USA: Lawrence Erlbaum Associates.

Teif, M., & Hazzan, O. (2006). Partonomy and taxonomy in object-oriented thinking: Junior high school students' perceptions of object-oriented basic concepts. *SIGCSE Bulletin*, *38*(4), 55–60.

Thomas, L., Ratcliffe, M., & Thomasson, B. (2004). Scaffolding with object diagrams in first year programming classes: Some unexpected results. In *SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (pp. 250–254). New York: ACM Press.

Truong, N., Bancroft, P., & Roe, P. (2005). Learning to program through the web. In *ITiCSE '05*: *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 9–13). New York: ACM Press.

Utting, I. (2006). Problems in the initial teaching of programming using Java: The case for replacing J2SE with J2ME. In *ITICSE '06: Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 193–196). New York: ACM Press.

Vainio, V., & Sajaniemi, J. (2007). Factors in novice programmers' poor tracing skills. In *ITiCSE '07: Proceedings of the 12th Annual Conference on Innovation and Technology in Computer Science Education* (pp. 236–240). New York: ACM Press.

Wiedenbeck, S., Ramalingam, V., Sarasamma, S., & Corritore, C. L. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with Computers*, *11*, 255–282.

Winslow, L. E. (1996). Programming pedagogy: A psychological overview. *SIGCSE Bulletin*, *28*(3), 17–22.

Authors' Note

All correspondence should be addressed to:
Jorma Sajaniemi
University of Joensuu
P.O. Box 111
FI-80101 Joensuu
Finland
saja@cs.joensuu.fi

Human Technology: An Interdisciplinary Journal on Humans in ICT Environments
ISSN 1795-6889
www.humantechnology.jyu.fi

Human Technology:
An Interdisciplinary Journal on Humans in ICT Environments

www.humantechnology.jyu.fi