
Teaching Novices Online: Does Presentation Order Matter?

Douglas A. Kranch
Computer Information Systems
North Central State College
2441 Kenwood Circle
Mansfield, OH 44906
(419) 755-4788
dkranch@ncstatecollege.edu

Abstract

Studies of expertise development have found that novices begin by learning surface features, progress to competence, and then undergo a unique reorganization of their knowledge before becoming experts. This study directly compared three instructional sequences presenting identical content to different groups of programming novices: beginning with basic skills and progressing in a hierarchical order; beginning with abstract plans for novices to match to problems; and spiraling through both basic and abstract topics. Learning was measured by performance on a Programming Assessment given immediately after the instruction, and effort and difficulty were self-rated during the instruction. There was no significant difference among the groups in Programming Assessment scores, and overall self-rated effort and difficulty of the instruction did not vary simply by rearranging the presentation order of the major elements. However, instructional units that covered programming syntax skills and structures were rated by all groups as requiring significantly less effort and difficulty than units covering plans, and participants in all groups scored significantly higher on syntax skills and structures than on plans. These results help identify the order in which introductory programming instruction should be presented to novices for maximum learning efficiency: syntax first, then structures, and finally programming plans. By extension, novices do not benefit by starting them with overviews and strategies, but with fundamentals.

Instructors, with their years of expertise, may feel the urge to present to novice students their accumulated body of knowledge in a way that “jump starts” them on the road to expertise. They may do this through topic organizers, overviews, and “big picture” strategies that they hope novice students will use to guide them through their initial steps in the domain. But does this kind of high-level overview make a difference to novices’ learning?

Review of the Literature

Expertise from Mental Organization and Perception

The knowledge structure that ultimately forms within an expert is a direct result of the memory structure of the human brain. Cowan (2000) saw strong evidence for dividing human memory into a long-term and a short-term component. Based on his comprehensive review of memory studies, Cowan (2000) estimated the capacity of short-term memory at four elements when mnemonic devices are discounted. For expertise to develop, the limited capacity of short-term memory must be overcome using the mind’s other resources, principally perception. By enhancing perception, what one sees and the way one sees it can be fundamentally changed.

Such a prioritized shift in perception as a means to increase efficiency goes to the heart of expertise. For example, Boschker, Bakker, and Michaels (2002) found that expert wall climbers were better able to identify usable wall holds in the upper and middle areas of the climbing wall than novices. Experts further characterized these affordances as coming in varying levels of detail or grain with affordances of coarser grain emerging from those of finer grain. The expert climber's mind used this perceptual organization as a recall mechanism, each higher level affordance a collection of lower level affordances that allowed easier retrieval of the entire climbing wall.

The source of this heightened perception lies within the perceiving mind. Miller (1956) in a study comparing the discrimination ability of the senses determined that the human mind -- enhances perception by distilling sensory information into 7 ± 2 (i.e., from 5 to 9) different natural levels. Chase and Simon (1973) explored the ability of chess experts to recall pieces on a chessboard and found that the number of pieces chess experts could correctly recall was far above the Miller's seven-element limit. They concluded that the experts' long term memories must have organized conceptually-related groups of pieces into clusters that they named chunks, so that the number of total pieces the experts' short-term memories could perceive at one time became the sum of all the pieces contained in all the chunks. Thus, expertise means mastering the complex by making it simple. Rather than *dividing* and conquering a domain, in overcoming the limitations of short-term memory the mind *combines* and conquers it.

It is common to ask people to "think outside the box" when a seemingly intractable problem is encountered. Experts, on the other hand, spend many years studying their field to build such a "box" around it that for them makes the field both predictable and controllable. It seems that chunks are the means by which the attention of short-term memory is sharpened, increasingly leading experts to anticipate the elements they expect to perceive within the domain and so build their "box."

Acquiring Expertise

Zeitz and Spoehr (1989) focused on this evolving internal knowledge organization as a three-stage process. The novice in the first stage learned to recognize perceptual features and then gradually inculcated them into long-term memory as chunks. As the essential components of the chunks themselves were still being learned and interconnected, learning was by rote, organization was poor, and the chunks were difficult to recall. In the second stage, the basics of the lowest level chunks had been mastered as well as the organization of the domain as presented. The learner now had internalized a cognitive replica of the content and organization of the domain. In the third stage, continued interaction with the domain reworked the cognitive organization by adding unique content within the chunks and unique horizontal and vertical associations among the chunks to produce complex, personalized, and difficult to verbalize expertise.

Developmental stages in expertise suggests that the instruction used at each stage must be tailored to the special strengths and challenges of each stage. Novices are focused on surface features and rules. They want step-by-step solutions and tend to memorize specific solution algorithms to match the surface feature of the presented problems. They also need to have filled vast holes in their knowledge about the domain, and that domain is generally well defined with fully developed facts, principles, and algorithms. An additional complication is that novices come to the new domain with a variety of knowledge and skills, some relevant to the domain.

Novices are best served by a direct approach to instruction, a model that takes into account differences in entry skills and aptitudes, is efficient, and is focused on achieving discrete learning objectives based on skill mastery (Huitt, Monetti, & Hummel, 2009). Using a direct approach, novices have a ready-made mental structure of the domain on which to build their competence.

Developing Programming Expertise

This study focused on the programming domain. Previous programming studies (Adelson, 1981; Bonar & Soloway, 1985; Corritore & Wiedenbeck, 1991; Davies, Gilmore, & Green, 1995; Spohrer & Soloway, 1988; Wiedenbeck, (1986) showed that novices focus on learning surface features that let them differentiate among the varieties of syntax and semantic elements each language offers. Their approach to program problem solving is to match those -surface features to problem requirements. As programming expertise develops, attention gradually shifts from the surface features to the functional features of a programming language. Ultimately, programmers depend less on backward and bottom-up designing from scratch and more on top-down forward design based on schemas retrieved from memory and adapted to the goal as needed (Rist, 1989).

It is these schemas and strategies that experts, now instructors, see as the key to their expertise and which they want to teach to their novice students as they explain to them how they themselves write programs. But their own programming expertise resulted from years of acquiring their skills and knowledge as chunks which they combined and organized as themes were discovered. Can plans and strategies that took years of study and practice to master be taught directly to novices? If so, in what sequence?

Several instructional sequences have been suggested. Like many expert teachers, Soloway (1986) would begin with generic plans for solving problems and teach students to modify and combine those plans to solve problems. On the other hand, Gagné, Briggs, and Wager (1988) would sequence skills in a hierarchical fashion so that all prerequisite skills are acquired before subsequent skills are attempted. Reigeluth and Stein (1989) would center the sequence on epitomes that would be gradually elaborated. Knowing the answer to this question is especially important in online instruction as learners are seldom given a choice of sequences for presenting the content to be learned.

Methodology

This study employed a three-group experimental design to examine which of the three instructional sequences would best improve the programming skills of novice programmers. A Web-based instructional program designed as a series of Web pages was used that covered the basic programming structures of sequence, selection, and repetition in Visual Basic. To provide uniform content, this one program was arranged in three sequences: an *elements-to-plans* (or *elements*) sequence based on Gagné, Briggs, and Wager's (1988) hierarchical skill sequencing, a *plans-to-elements* (or *plans*) sequence derived from Soloway (1986), and a *spiral* sequence modeled after Reigeluth and Stein (1983). Each sequence was divided into three units to provide two short breaks and so avoid overtaxing the participants.

The design of a basic presentation page is shown in Figure 1. Navigation links were provided at both the top and bottom (not shown) of each page. The text in each page was set to present a single topic without requiring the learner to scroll. The Web page shown in Figure 1 is typical of pages focused on the syntax of the language, in this case explaining the rules for using data types.

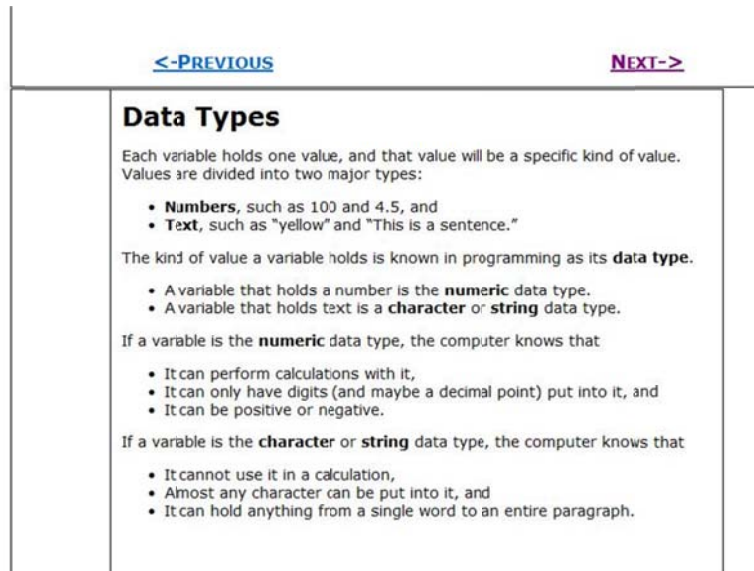


Figure 1. Layout of a Web page used for syntax units in the instructional sequences.

Two types of interaction were provided. The first allowed the participant to insert values into text boxes and select an operator in the programming statement at the top of the page. When the participant clicked the "Run" button, the statement would be executed using the values inserted by the participant. Questions that let the participants check their understanding of the concepts were provided at the bottom of most pages. Free play with practice programs was encouraged to give participants experience with structures without requiring the frustration of writing and correcting the entire code. They could enter limit values and operators to see the behavior of the structure without concern that unexpected results were due to an error in the program. The units discussing program plans provided text boxes for participants to use to write their answers to the self-test questions.

The study sample ($N = 34$) was recruited by personal solicitation from entering nursing and Computer Information Systems (CIS) students. These two groups were used because it was assumed that their career interests and preparation would give them sufficient math background to learn programming successfully. In addition, nursing students were recruited to represent students who were naïve to computer use beyond word processing and other widely-used applications. Participants were randomly selected to use one of the three sequences. To insure that the groups were statistically equivalent, measures were taken of general mathematical competence, programming self esteem, and hours spent playing computer games, factors shown by Bergin and Reilly (2006) to be important preconditions to success in programming courses.

Mental effort and mental difficulty were self-rated on a 9-point scale by the participants at the end of each of the three instructional units in each sequence, with 1 being the lowest rating and 9 the highest. Mastery of the syntax, structure, and plan skills taught in the units was measured with a Programming Assessment divided into three sections. The first section of nine items assessed participant knowledge of programming syntax elements, the second of six items participant knowledge of programming structures, and the third of ten items participant knowledge of

programming plans. Table 4 shows the mean percent of questions correct for each question type for all three instructional sequences.

Table 4. Descriptive Statistics for Programming Assessment Subtotals

Subtotal	N	Mean	Std. Dev.	SEM
Syntax Subtotal Percent	34	60.6862	22.73545	3.89910
Structure Subtotal Percent	34	45.7838	22.40216	3.84194
Plans Subtotal Percent	34	30.8235	22.08588	3.78770

Results and Discussion

Learning versus Effort

Participants as a group received significantly lower scores ($p < .001$) on average on the Programming Assessment as they passed from the syntax to the structure and finally to the plan items with no significant difference in item scores among the three learning sequences, indicating all learning sequences resulted in a similar drop in score as participants moved from the simpler syntax items to the more complex plan items. No learning sequence produced a significantly better performance on the assessment as a whole or on any part of it. Rather, all learning sequences produced the same drop in score as participants moved from the simpler syntax items to the more complex plan items. All of the instructional sequences in the study appeared to yield equal development of programming expertise for novice programmers.

When the three unit measures of effort and difficulty are averaged over the entire instruction, an ANOVA by instructional sequence reveals no significant difference among the sequences for mean effort ($p = .152$) or mean difficulty ($p = .261$). When the units are examined individually, however, a different picture emerges. Figure 2 shows graphically the mean unit effort ratings for the three learning sequences and, hence, the intrinsic load produced by each unit (DeLeeuw & Mayer, 2009). Unit 1 of both the elements and spiral sequences contained content related to syntax elements and were rated at about 5, but the remaining spiral sequence units saw intrinsic load increase rapidly, while the elements sequence did not have a significant increase in intrinsic load until Unit 3. The plans sequence began with high intrinsic load in Unit 1, which contained exclusively strategic planning content, then dropped significantly for the remaining units that contained syntax elements and structure content. For all three sequences, intrinsic load was high for units containing content related to plans as reflected by a mental effort rating of 7 or more, while it was near the ideal middle level of 5 for units with content related to syntax elements. Thus, syntax elements material was consistently perceived as requiring less effort and as less difficult than program plan material.

The self-rated difficulty scores shown in Figure 3 show almost the identical mean ratings by sequence and unit as self-rated effort. The difficulty ratings indicate that more germane load was experienced in units with program plan content than unit with syntax elements content. There

was also a high degree of correlation between the effort and difficulty ratings for all three units ($p < .001$).

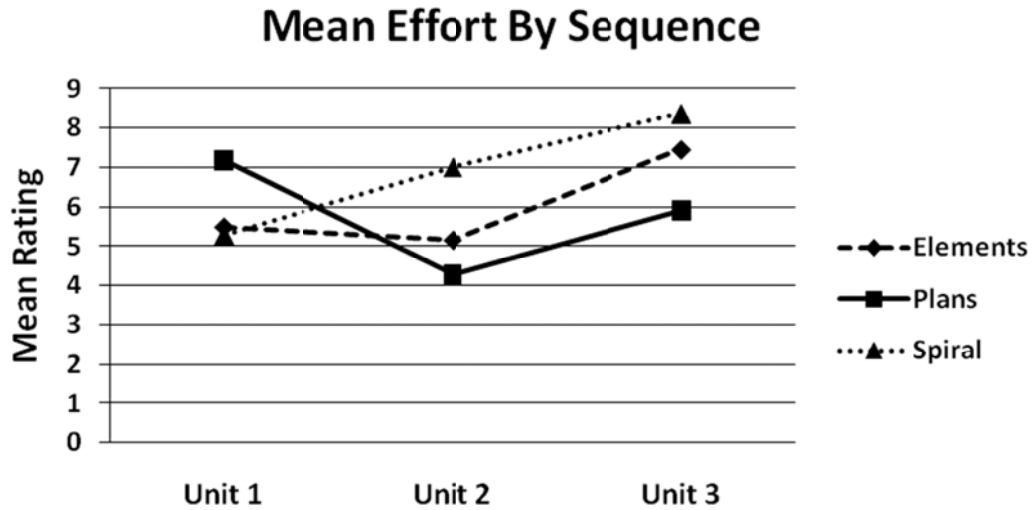


Figure 2. Mean effort reported by participants in the elements, plans, and spiral sequences for the three instructional units.

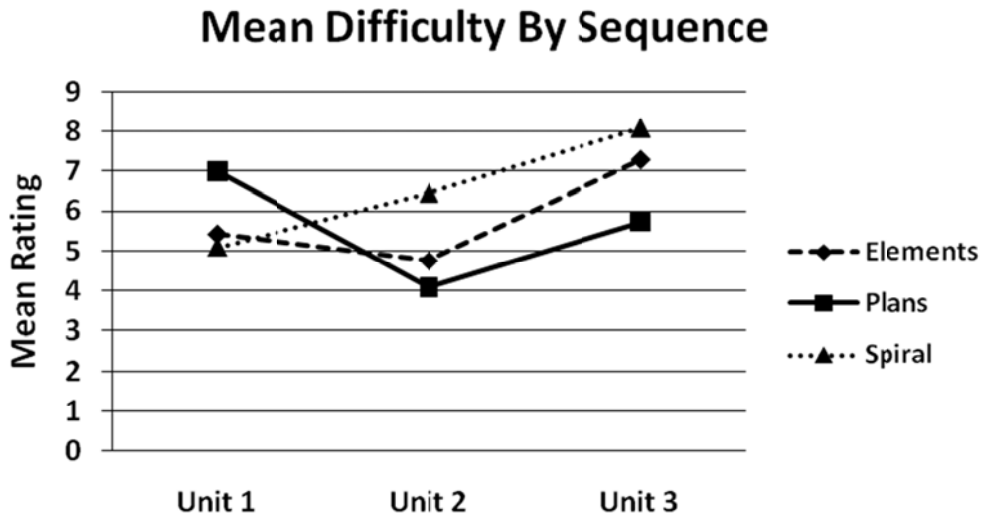


Figure 3. Mean difficulty reported by participants in the elements, plans, and spiral sequences for the three instructional units.

Difficulty Unaffected by Sequence

Evidence of a relationship between content and effort/difficulty can be seen by comparing the mean effort scores for the elements and plans sequences. Both sequences contained identical material in their units but presented in opposite order. The elements sequence progressed from syntax elements to structures to plans, while the plans sequence progressed from plans to structures to syntax elements. The content of unit 2 in both sequences was identical. Figure 4 presents the

same effort means as Figure 3 but with the effort means for units 1 and 3 reversed for the plans sequence so that units with identical content are placed in the same relative locations on the graph. The similarity of the means for the two sequences is clearly visible, and there is no significant difference between effort or difficulty scores for any of the units in the elements and plans sequences. Perceived difficulty and effort followed the content no matter which sequence was used. These nearly identical results also rule out the mental tiredness as a cause for increased effort and difficulty as program plans were covered last in the elements sequence and first in the plans sequence.

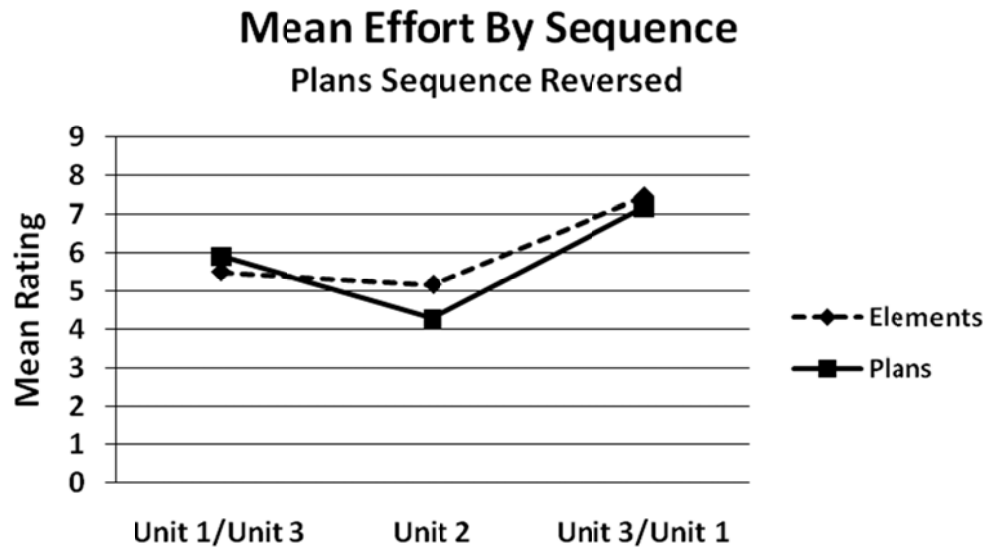


Figure 4. Mean effort reported by participants in the elements and plans sequences for the three instructional units, with effort in units 1 and 3 reversed for the plans sequence.

Effort and difficulty, though measuring the different constructs of intrinsic load and germane load (DeLeeuw & Mayer, 2009), are tightly intertwined. Units containing syntax elements were significantly lower in effort and difficulty than units containing planning content, a pattern that was evident no matter how the content was sequenced. This result supports the contention that novices find working with plans and strategies much more difficult than with the basic elements of syntax and structure.

While the units in which the complex materials was presented were rated individually as higher in effort and difficulty, none of the strategies used to present the complex material moderated overall the participants' ratings of effort or difficulty for the instruction. Averaged over all three units, effort and difficulty were rated the same by participants in the three instructional methods, showing that changing the order in which easier or more challenging information was encountered did not alter the perceived effort or difficulty of the material. Neither moving the difficult material nor increasingly contextualizing it resulted in any change in participant ratings.

Difficulty Dependent on Content

There were, however, significant differences among the overall mean scores for different content items, with participants on average achieving the highest scores on the syntax items, middle

scores for structure items, and lowest scores for plan items. The implication is that novice programming learners master more of the foundational material than the advanced material regardless of the order in which the material is presented to them, whether from simple to complex, complex to simple, or intermixed. Since novice learners focus first on the surface characteristics, syntax elements should be easiest for them to learn, with feature-laden structures nearly as easy. Participants appeared to focus on and learn the same material regardless of the order in which they were presented, to selectively attend to the material and block out any that was not understood. Where there was no understanding, it seems there also was neither attention nor chunk formation.

Selective attention to content did not come without a price. The effort and difficulty ratings of the elements and plans sequences both indicated that the units containing syntax and structure material were perceived to be easier than the unit containing programming plan content. The spiral sequence, on the other hand, had programming plan content distributed through all three units and they were all rated at about the same high level of effort and difficulty. This may reflect the increased cognitive load required to separate the syntax and structure material from the programming plan material and subsequently attend to the former and ignore the latter (Chandler & Sweller, 1991). If material that is out of scope of the learner's current knowledge is acknowledged to be extraneous for that learner, then slowing learning without having any effect on cognitive organization is a plausible result.

The plans sequence discussed programming plans first and seemed to experience detrimental effects from it. Soloway (1986) advocated teaching novices programming plans from the beginning as a way to jump-start their expertise; however, participants learning from the plans sequence did not seem to benefit from this. Rather, the entire learning sequence was described as seeming to "jump around" with no apparent order. Such confusion was not evident in the comments about the spiral sequence, which mixed programming plans throughout all its units in the simple-to-complex spiral presentation recommended by Reigeluth and Stein (1983). It may be that a little incomprehension is tolerable to learners but that, once a threshold of incomprehension has been crossed, learners pass a summary judgment on the entire instructional sequence.

Conclusion

What do these findings mean for teaching novices, especially in online courses? Two important design guidelines can be drawn from them. First, it is critical to determine what novice learners can comprehend in the domain. The novices in this study demonstrated little learning of topics beyond what made sense at their level. In the same way, online learners will pass over topics too advanced for them with little or no comprehension. Second, learners do not just ignore topics for which are not ready; they expend increased effort to learn what they can understand, which simultaneously increases the difficulty of the lesson. The effort and difficulty may be so great that the learners conclude that the lesson is simply too difficult to continue. This is where thorough knowledge of the domain and thorough field-testing of the online units become critical. Rather than providing an advance organizer to the field or shortcuts to competence, the novices in this study found high-level summaries and strategies confusing and the most difficult to learn. By extension, novices do not benefit by starting them with overviews and strategies, but with -- domain fundamentals. The most accurate way to determine what topics novices are ready to learn is by measuring accurately what they have learned and letting them rate the effort they expend and difficulty of the lessons. Doing this will go a long way to ensuring that online lessons are well suited for novice learners.

REFERENCES

- Adelson, B. (1981). Problem solving and the development of abstract categories in programming languages. *Memory & Cognition*, 9(4), 422-433.
- Bergin, S., & Reilly, R. (2006). Predicting introductory programming performance: A multi-institutional multivariate study. *Computer Science Education*, 16(4), 303-323.
- Bonar, J., & Soloway, E. (1985). Preprogramming knowledge: A major source of misconceptions in novice programmers. *Human-Computer Interaction*, 1(2), 133-161.
- Boschker, M. S. J., Bakker, F. C., & Michaels, C. F. (2002). Memory for the functional characteristics of climbing walls: Perceiving affordances. *Journal of Motor Behavior*, 34(1), 25-36.
- Chandler, P., & Sweller, J. (1991). Cognitive load theory and the format of instruction. *Cognition & Instruction*, 8(4), 293-332.
- Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology*, 4, 55-81.
- Corritore, C. L., & Wiedenbeck, S. (1991). What do novices learn during program comprehension? *International Journal of Human-Computer Interaction*, 3(2), 199-222.
- Cowan, N. (2000). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24(1), 87-114.
- 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(2-3), 227-248.
- DeLeeuw, K. E. & Mayer, R. E. (2009) A comparison of three measures of cognitive load: Evidence for separable measures of intrinsic, extraneous, and germane load. *Journal of Educational Psychology*, 100(1), 223-234.
- Gagné, R. M., Briggs, L. J., & Wager, W. W. (1988). *Principles of instructional design* (3rd ed.). New York: Holt, Rinehart and Winston..
- Huitt, W. G., Monetti, D. M., & Hummel, J. H. (2009) Direct approach to instruction. In C. M. Reigeluth & A.A.Carr-Chellman (Eds.), *Instructional-design theories and models volume III: Building a common knowledge base* (pp. 334-381). New York:Taylor and Francis.
- Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63, 81-97.
- Reigeluth, C. M., & Stein, F. S. (1983). The elaboration theory of instruction. In C. M. Reigeluth (Ed.), *Instructional-design theories and models: An overview of their current status* (pp. 334-381). Hillsdale, NJ: Lawrence Erlbaum Associates.

- Rist, R. S. (1989). Schema creation in programming. *Cognitive Science*, 13, 389-414.
- Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM*, 29(9), 850-858.
- Spohrer, J. C., & Soloway, E. (1988). Novice mistakes: Are the folk wisdoms correct? *Communications of the ACM*, 29(7), 624-632.
- Wiedenbeck, S. (1986). Beacons in computer-program comprehension. *International Journal of Man-Machine Studies*, 25(6), 697-709.
- Zeitz, C.M., & Spoehr, K.T. (1989). Knowledge organization and the acquisition of procedural expertise. *Applied Cognitive Psychology*, 3(4), 313-336.