The Effect of Dynamic Copies Model in Teaching Recursive Programming

Ming-Puu Chen Ph.D.
Department of Information and Computer Education
National Taiwan Normal University

The copies model of recursion was implemented in two versions of computer-based instruction (dynamic vs. static) in this study. For the immediate effects, dynamic copies model was more effective than static copies model in teaching recursion. High prior knowledge students performed better than low prior knowledge students no matter they were instructed with the dynamic or static copies model. For the delayed effects, ATI was found. High prior knowledge students benefited from the static copies model instruction more than from the dynamic copies model. In contrast, low prior knowledge students benefited from the dynamic copies model more than from the static copies model.

keywords: Computer-based instruction Conceptual model Computer programming Cognitive load

Introduction

Statement of the Problem

Recursion is one of the most complex programming constructs for students to learn and for teachers to teach (McCrank, 1987). Most novice programmers encounter great difficulty in comprehending recursive repetition even after an introductory university computer programming course (Kessler & Anderson, 1989; Pirolli & Anderson, 1985; Wiedenbeck, 1988). Anderson, Pirolli and Farrell (1988) have found that novices have great difficulty in learning recursive functions due to the unfamiliarity of recursion. There are two reasons for this. First, novice programmers usually lack a suitable mental model for comprehending recursive flow of control. Second, recursion, which is generally compounded by variable binding, parameter passing, and environmental closure, is too complex for novice programmers (Er, 1984). Anzai and Uesato (1982) have concluded that learning iteration first helps with recursion because iteration provides a model for a recursive call. Learning recursion first, however, does not help with iteration because iteration is easily understood, and the recursive construct is not understood well enough to serve as the basis for transfer. Effective instructional methods which provide appropriate cognitive support are needed in facilitating learning recursion.

Background

Conceptual model. Novice programmers usually possess inappropriate mental models and misconceptions about computers and programming languages (Bayman & Mayer, 1983, 1988). Mayer's (1981, 1982, 1985, 1987, 1988) series of programming research have suggested that instruction with conceptual models reduces misconceptions, enhances mental models, and enhances problem-solving. Many conceptual models, such as the nested Russian dolls, the inductive model, the analogy model, the graphical recursive structure model, the mathematical model, the tree model, and the
copies model have been implemented to facilitate the comprehension of recursive process (Murnane, 1991; Wilcocks & Sanders, 1994).

A conceptual model of recursion is the conceptualization of recursion held by expert programmers and provides an appropriate representation of recursion, which is appropriate in the sense of being accurate, consistent, and complete (Norman, 1983; Mayer, 1989a). Conceptual models can be used to facilitate a learner's development of an accurate mental model of recursion. Less able novice programmers perform better if they are given a concrete model before learning recursion (Mayer, 1981). In contrast, able novices do not improve their performance as a result of being given such a model. Pirolli and Anderson (1985) have concluded that successful learning of recursion depends on an adequate mental model of recursion. Kessler and Anderson (1989) have also argued that possessing an adequate mental model was also critical to the transfer of learning between recursion and iteration. Having a model of iteration enables novices to see how to transfer their knowledge to recursion. In contrast, novices studying recursive programming without the aid of such models are overwhelmed by the surface differences between iteration and recursion and start out with means-end analysis in solving the problem (Kessler & Anderson, 1989). Therefore, how to help novice programmers to develop adequate mental models of recursion is a critical factor in teaching and learning recursion.

The underlying difference between experts and novices lies in their respective degrees of conceptual understanding (Payne, 1988). With the assistance of conceptual models, novice learners can develop better mental models and thus perform tasks more like experts (Borgman, 1986; Mayer, 1988, 1989a; Payne, 1988). Norman (1983) have found that using conceptual models enables analogical learning and cognition. When learners acquire useful conceptual knowledge such as conceptual models, they can use this knowledge for learning and thinking about a related new domain.

Simplicity and visibility are two important characteristics for novice programmers to learn computer programming (Du Boulay, O’Shea, & Monk, 1981). The glass box approach conceptual model suggested by Du Boulay, O’Shea and Monk (1981) offers the novice learner a view of the internal operations of a computer and the way the system reacts to programs. The level of detail, however, must be sufficient to illustrate the target concept and should not be too complex to interfere the understanding of the concept. Visual conceptual models have been suggested by Shih and Alessi (1993) to foster learning computer programming.

**Worked-example.** Instructional material that promotes learning by example typically contains numerous worked-examples that learners study and use as models for solving problems. A worked-example consists of a problem and an explicit sequence of steps to a correct solution. The learning-by-example approach is popular in computer manuals, which illustrates features and commands with examples of how they can be applied in specific situations. Reder, Charney, and Morgan (1986) have found that computer users perform on-line tasks better after studying manuals with worked-examples rather than unelaborated manuals.

After reviewing previous studies on learning computer programming, Pirolli and Anderson (1985) have summarized that studies of how people learn to program all emphasize the importance of analogical reasoning as a basis for learning to program. The analogical reasoning process can be characterized as a complex mapping procedure that goes beyond mere copying of an example. In related studies (Anderson, Farrell, & Sauers, 1984; Carbonell, 1983; Gentner & Gentner, 1983; Gick & Holyoak, 1983; Pirolli & Anderson, 1985), learners can make use of examples to write syntactically correct code and develop adequate mental models of programming construct from analogies. In contrast, learners who do not develop an adequate mental model of programming construct use a mapping strategy that heavily relies on the surface features of the examples. Novices learning from worked-examples also demonstrate great savings in solution time when going from the first problem to the next. Worked-examples help learners categorize problems with similar solutions and construct solutions to novel problems by analogy to the example (Anderson, Farrell, & Sauers, 1984; Sweller & Cooper, 1985). Pirolli and Anderson (1985) have found that learners show less reliance on the example in subse-
quent problems. Worked-examples reduce errors on the first opportunity for acquiring a skill but does not affect subsequent rates of improvement. Worked-examples with verbal explanation of structure reduce training time when compared with instruction which only presents the explanation of the processes of computer programs (Pirolli, 1991).

Copies model of recursion. There are several possible conceptual models that can be considered to represent an understanding of recursion: the analogy model, the graphical recursive structure model, the mathematical induction model, the tree model, and the copies model (Wilcocks & Sanders, 1994). In the copies model approach, a recursive procedure can be understood in terms of a procedure looping over a stack of function calls (Kahney & Eisenstadt, 1982; Kessler & Anderson, 1989). Recursion is a process that is capable of triggering new instantiations of itself, with control passing forward to successive instantiations and back from terminated ones. The copies model of recursion is the model assumed to be possessed by expert programmers (Kahney, 1989) and can be easily represented as a series of overlapped windows by dynamic media such as a computer screen (Wilcocks & Sanders, 1994).

Purpose and Rationale

This study was designed to investigate the effect of using dynamic copies model as instructional cognitive support on learning recursive programming and how instruction may interact with prior knowledge. The design of this study was based on the cognitive load theory (Chandler & Sweller, 1991; Sweller, 1988, 1989, 1993, 1994, 1995) and related assumptions. The copies model of recursion and worked-examples discussed in the previous section were used in implementing the instructional material for this study.

Cognitive load theory. Cognitive load theory has been used to suggest that many commonly used instructional procedures and materials are inadequate in the manner which impose heavy cognitive load to the learner. This theory is based on the following assumptions concerning basic cognitive constructs:

1. People have a very limited working memory which can hold and process only a few items of information at a time.
2. People have an extensive long-term memory that is effectively unlimited in size.
3. Schema acquisition is a primary mechanism of learning.
4. Automation of cognitive processes is a learning mechanism that bypasses working memory and reduces working memory load.

Cognitive load imposed by task-information may exhaust cognitive resources of individuals and consequently interfere learning (Halford, 1993). According to cognitive load theory, learning frequently fails not because of the intrinsic complexity of the information, but rather because it is presented in a manner that requires learners to engage in unnecessary cognitive activities. Sometimes, because of the format, instructional material must be mentally integrated or restructured before learning taking place (Sweller & Chandler, 1994). Therefore, unnecessary cognitive activities must be engaged in the learning process, not because they are essential to learning but because the information is not appropriately presented. The engagement in unnecessary cognitive activities imposes a heavy working memory load on the learner and interferes learning.

According to cognitive load theory, conventional problem solving which emphasizes goal attainment leads novice learners to use means-ends analysis to search for differences between the problem state and the goal state. This misdirects learners' attentions and imposes a heavy cognitive load. For learning to occur, attention needs to be directed to each problem state and its associated solution moves, not to a cognitively demanding search process for relevant operators or differences between problem state and goal state. The means-ends analysis involves working backward from the goal, which overloads processing capacity and leaves few resources for schema acquisition (Sweller, 1988, 1989). Thus, an implication of cognitive load theory is that an increased emphasis on goal attainment during learning actually decreases schema acquisition. Alternative instructional procedures, such as goal free problems (Owen & Sweller, 1985), worked-examples (Pillay, 1994), mixing auditory and visual presentations (Mousavi, Low, & Sweller, 1995), and integrated technical illustrations
(Purnell, Solman, & Sweller, 1991), have been implemented based on cognitive load theory to reduce extraneous cognitive load.

However, there is apparent discrepancy between constructivist theories and the limited capacity theories, such as the cognitive load theory (Goldman, 1991). The research finding of learning additional material improved performance despite an increase in cognitive load is inconsistent with the cognitive load theory (Dixon, 1991). Similarly, Goldman (1991) have argued that a larger investment of initial effort in the task sometimes benefits in longer term.

**Split-attention effect.** The split-attention effect occurs when learners are required to divide their attention among and mentally integrate multiple sources of information such as statements and associated diagrams (Mousavi, Low, & Sweller, 1995; Sweller & Chandler, 1991, 1994). A physically integrated format reduces the load on working memory. Learning that requires learners to mentally integrate multiple sources of information may result in less effective acquisition of information than if the information is presented in an integrated form. Ward and Sweller (1990) have found that learning from worked-examples that requires the learner to mentally integrate disparate sources of information is not better than solving the equivalent problems.

A major reason for the ineffectiveness of some worked-examples is that they impose a heavy extraneous cognitive load that may not differ from that imposed by a means-ends strategy (Cooper & Sweller, 1987). The act of mental integration involves searching relations among elements associated with the diagram and statements. Searching relations among disparate elements requires cognitive resources and is irrelevant to schema acquisition. The extraneous cognitive load is imposed because of the manner in which the material is presented. The cognitive load consequence of split-attention, however, can be overcome by physically integrating multiple sources of information (Chandler & Sweller, 1991, 1992; Ward & Sweller, 1990).

**Individual difference in prior knowledge.**

Individual differences among students present a pervasive and profound problem to educators. These differences somehow condition students' readiness to profit from particular instructional environments (Snow, 1986). Corno and Snow (1986) have concluded that effective instructional methods serve to circumvent inaptitude by taking over the relevant cognitive or behavioral burdens for low aptitude learners. Clark (1989) also have argued that instructional methods must not impose less effective or less desirable cognitive processing strategies on higher aptitude students. Instructional methods must go beyond simply providing information and support cognitive processing when students are unable or unwilling to do so for themselves (Clark, 1990).

Individual differences in prior experience have been found to affect the performance and attitude of users of computer programs (Kieras & Polson, 1985; Vicente, Hayes, & Williges, 1987). Previous studies (Aman, 1992; Clarke, 1992; Carlson & Wright, 1993) have suggested that the most reliable predictions of computing attitude and achievement are based on the amount of prior computing knowledge. Weinert, Helmke and Schneider (1989) have found that prior knowledge is either a necessary or at least a facilitating factor in the acquisition of new knowledge in the same content domain. Individuals who have greater prior knowledge will learn more quickly and more effectively. Evidence indicates that it is the domain-specific knowledge difference that enables experts to recognize so many domain-specific patterns and to automatically apply this knowledge to solve domain-related problems efficiently (Siegler, 1982).

The domain-specific expertise is the most important difference between novices and experts in various knowledge domains, such as physics (Chi, Glaser & Rees, 1982), algebra (Lewis, 1981), geometry (Anderson, Greeneo, Kline, & Neves, 1981), and computers (Howerton, 1988; Taylor & Mounfield, 1994). Experts and novices typically do not differ with respect to general strategies or working-memory size, but do differ significantly in both the quantity and the quality of domain-specific knowledge they possess (Chi, Glaser, & Rees, 1982). Sternberg (1981) have suggested that the ability of the more able people to organize their knowledge in a domain allow them to access and use this knowledge efficiently and effectively. Weinert, Helmke and Schneider (1989) have found that do-
main-specific knowledge can explain learning achievement better than intelligence, and the differences between individuals' prior knowledge can be reduced by instruction. In classes with intensive individualized support, perceived ability is a much less important determinant of math achievement than it is in classes where teachers give students little individual support. Mayer and Sims (1994) have argued that prior knowledge compensates for poor instruction. In other words, both high and low prior knowledge students benefited from good instruction, but when receiving poor instruction, high prior knowledge students outperform those with low prior knowledge.

Many prior knowledge studies have been done in various knowledge domains. Few studies, however, have been done in the area of computer programming, especially in learning recursion. Whether domain-specific knowledge enhances learning recursion from conceptual models needs to be studied in more detail.

**Methodology**

**Subjects**

There were 258 high school students from six classes participated in this study. Three classes of the participants were level one students and the other three classes were level two students. The three classes of level one participants were taking the BASIC programming language when this study was conducted. The three classes of level two participants had already taken the programming language course in the last academic year. No subjects learned recursive programming before. Therefore, all subjects were considered as novices to recursive programming. The experiment was implemented as a computer-based supplemental instruction for all participating classes. Participants were instructed with a computer-assisted instruction program on IBM compatible personal computers during regular course meetings.

**Experimental Design and Procedure**

The design of this study was an aptitude-treatment-interaction (ATI) study. Subjects' test scores were tested as continua instead of categorical data. The independent variables were the type of copies model (dynamic vs. static) and prior knowledge. The dependent variables (criterion variables) were subjects' performance in recursive tasks measured by the immediate posttest and the delayed posttest.

Two weeks before the treatment, participants were randomly assigned to either the dynamic copies model group or the static copies model group and all participants completed a domain knowledge test on prior knowledge of computer programming language. The content of the domain knowledge test consisted of basic computer concepts, variable assignment, control structures, iteration, and recursion to identify subjects' prior knowledge of computer programming and to examine whether some participants were already familiar with recursive programming.

In the treatment week, the immediate posttest was conducted to evaluate subjects' understanding of recursion immediately after the treatment. Two weeks after the treatment, the delayed posttest was conducted to evaluate subjects' understanding of recursion.

**Instrumentation**

There were three kinds of instruments used in this study: (a) the domain knowledge test, (b) two achievement tests on recursion, and (c) two sets of computer-based instruction.

**Domain knowledge test.** The purpose of the domain knowledge test was to identify subjects' prior knowledge before learning recursion. The domain knowledge test used in this study was a review test used by the teachers, which includes sixteen questions on control structures, iteration, and two questions on recursive functions. The internal consistency reliability of the domain knowledge test was .74 and the validity of the domain knowledge test was .65 which was the correlation coefficient between the test scores and the final scores of the programming course. The item difficulties ranged from .10 to .95 with an average difficulty of .45. The test was considered to be a reliable instrument.
based on the reliability, validity, and item difficulty.

**Two achievement tests.** Two twenty-minute recursion achievement tests, the immediate posttest administered immediately after the treatment and the delayed posttest administered two weeks after the treatment, were developed for this study. The purpose of the achievement tests was to evaluate students' understanding of recursion and to examine whether the instructional goals of the treatment instruction had been achieved. Therefore, two types of questions, code evaluation: predicting the results of a given recursive program and code generation: generating the recursive solution to a given problem, were included in the achievement tests to examine whether the instructional goals of the treatment instruction have been achieved.

The internal consistency reliability of the achievement tests were .82 and .80 for the immediate posttest and the delayed posttest, respectively, as measured by Cronbach's coefficient alpha. The validities of the achievement tests were .43 and .40 for the immediate posttest and the delayed posttest, respectively, which were the correlation coefficients between the achievement test scores and the final scores of the programming course.

**Two sets of computer-based instruction.** The treatment for this study was the use of dynamic copies model of recursion. Two versions of computer-based instruction on recursive programming were developed based upon cognitive load theory and worked-example research to provide cognitive support and reduce cognitive load. The computer-based instruction was conducted in the treatment week in a regular fifty-minute course meeting. In spite of the presentation format of the conceptual model, the two version of computer-based instruction on recursive programming were identical. Both types of instruction used worked-examples and copies models as analogies to recursion. The static copies model instruction provided a static copies model along with step-by-step verbal explanation of the recursive process to enhance learning recursion. The dynamic copies model instruction provided a step-by-step computer animated copies model and verbal explanation to represent the dynamic characteristics of the recursive process to reduce cognitive load and enhance learning.

The copies model of recursion, which can be visualized as copies of the original recursion program, was used as the conceptual model to enhance learning recursion and was implemented in the computer-based instruction. Successful learning of recursion depended upon an adequate mental model. The copies model was the model which that expert programmers possessed. Therefore, the copies model of recursion was an adequate model in helping learning recursion. Furthermore, the copies model could be easily represented in either static or dynamic way on computer screens as a series of overlaid windows (Wilcocks & Sanders, 1994). Therefore, the copies model was the best choice for the purposes of this study.

The dynamic copies model, as shown in Figure 1 and 2, presented the recursive process of the worked-example as overlapped windows step by step along with the verbal explanation of the on-topped window. In other words, the dynamic model provided explanative text and explanative illustrations (Mayer, 1989b) to facilitate understanding of recursion. The static copies model, as shown in Figure 3 and 4, presented the entire recursive process of the worked-example as overlapped windows all at once with step-by-step verbal explanation indented with the explained window. In Mayer's words, only illustrations and explanative text were provided in the static copies model. Therefore, based on cognitive load theory, the copies model instruction was hypothesized to be more effective than the static copies model in teaching recursion. Based on previous research on prior knowledge (Mayer & Sims, 1994; Weinert, Helmke, & Schneider, 1989), the interaction between prior knowledge and the type of instruction was also hypothesized to be significant.
Figure 1. The dynamic copies model presented the recursive process as overlapped windows step by step along with the verbal explanation.

Figure 2. The dynamic copies model presented the recursive process as overlapped windows step by step along with the verbal explanation of the on-topped window.
Figure 3. The static copies model presented the entire recursive process as overlapped windows along with step-by-step verbal explanation indented with the described window.

Figure 4. Verbal explanation was presented step by step in the static copies model.
Data Analysis

The process to study ATI suggested by Pedhazur (1982) was followed to analyze the data for this study. Regression analyses were conducted by using the SPSS for Windows (Norusis, 1993) to examine whether interactions exist between treatment instruction and prior knowledge upon dependent measures. The criterion variables (dependent variables) for regression analyses were the immediate posttest scores and the delayed posttest scores. The independent variables (predictors) for regression analyses were the type of instruction and prior knowledge.

Analyses and Results

There were 258 students participated in this study and a total of 230 students (89% of participants) completed the domain knowledge test, the computer-based instruction on recursion, and both achievement tests. Twenty-eight participants who did not finish all the tests during this study were excluded from the analyses. The total cases for this study were 230 (N=230). The group means of prior knowledge are listed in Table 1. The mean score and standard deviation of the static copies model group were slightly larger than the dynamic copies model group. No significant difference, however, was found between two means, $F(1,229) = 3.1618, p = .0767$. Thus the two groups were equivalent on the prior knowledge test.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Group Means of the Prior Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
</tr>
<tr>
<td>For Entire Population</td>
<td>5.31</td>
</tr>
<tr>
<td>Dynamic Group</td>
<td>5.00</td>
</tr>
<tr>
<td>Static Group</td>
<td>5.62</td>
</tr>
</tbody>
</table>

Regression analyses were performed following Pedhazur’s (1982) suggestion of studying interaction to test hypotheses of the ATI design on the immediate effect and the delayed effect of the treatment. The Alpha level was .05 for regression analyses performed by using the SPSS for Windows (Norusis, 1993). The baseline regression model for this study was:

$$Y = b_0 + b_1 \text{GRP} + b_2 \text{PK} + b_3 \text{GRPxPK}$$

\(Y\): the immediate posttest or the delayed posttest

\(\text{GRP}\): type of instruction (dynamic model vs. static model)

\(\text{PK}\): prior knowledge of computer pro-

gramming

$$\text{GRPxPK} : \text{type of instruction} \times \text{prior knowledge}$$

According to Pedhazur’s (1982) suggestion of studying ATI, the above baseline model was evaluated first for the immediate effect and the delayed effect. In other words, the type of instruction \(\times\) prior knowledge interaction was evaluated in the first step of testing the baseline model. If significant interaction existed, then regression analyses were conducted for each treatment group. Otherwise, the interaction term was going to be removed from the model for subsequent analyses. Then, significant terms were kept and non-significant terms were removed until the separate regression equations were reached.

Analysis of the Immediate Posttest

The immediate effect of the treatment was assessed by conducting the immediate posttest which measured participants’ understanding of recursion immediately after the treatment. SPSS multiple regression analyses with the test method were employed to examine whether the interaction between type of instruction and prior knowledge or main effects existed in the immediate posttest. The group means of the immediate posttest are listed in Table 2. The dynamic group had a larger group mean and smaller standard deviation than the static group.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Group Means of the Immediate Posttest</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
</tr>
<tr>
<td>For Entire Population</td>
<td>13.05</td>
</tr>
<tr>
<td>Dynamic Group</td>
<td>14.29</td>
</tr>
<tr>
<td>Static Group</td>
<td>13.61</td>
</tr>
</tbody>
</table>
As shown in Table 3, the type of instruction × prior knowledge interaction was not significant, $F(1, 229) = 3.7200, p = .0550$, against the baseline model. The R-square was .3205, and the R-square change of the type of instruction × prior knowledge interaction was .0112. In other words, only 1.12% of variance in the immediate posttest was accounted for by the type of instruction × prior knowledge interaction. Because the type of instruction × prior knowledge interaction was not significant, the interaction was removed from the regression model for the subsequent analyses.

### Table 3: Summary of Regression Analysis on the Type of Instruction × Prior Knowledge Interaction for the Immediate Posttest

<table>
<thead>
<tr>
<th>Variable</th>
<th>B</th>
<th>SE B</th>
<th>β</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRP</td>
<td>-2.6620</td>
<td>.8832</td>
<td>-3.746*</td>
</tr>
<tr>
<td>PK</td>
<td>.2864</td>
<td>.0598</td>
<td>.2050*</td>
</tr>
<tr>
<td>GRP × PK</td>
<td>.2907</td>
<td>.1507</td>
<td>.4415</td>
</tr>
<tr>
<td>(Constant)</td>
<td>14.1254</td>
<td>1.4225</td>
<td></td>
</tr>
</tbody>
</table>

Note. $R^2 = .3205$, GRP = type of instruction, PK = prior knowledge. *$p < .05$.

After removing the type of instruction × prior knowledge interaction, the prior knowledge and type of instruction main effects were significant, $F(1, 229) = 98.6629, p < .0001$ for the prior knowledge, and $F(1, 229) = 8.2879, p = .0044$ for the type of instruction. The R-square of the main effects was .3093, indicating that 30.93% of variance in the immediate posttest was accounted for by prior knowledge and type of instruction main effects. The summary of regression analysis is shown in Table 4. The regression equations for the immediate posttest were:

- $Y_{\text{Dynamic PI}} = 10.5969 + 0.7387 \text{PK}$
- $Y_{\text{Static PI}} = 9.4610 + 0.7387 \text{PK}$

As shown in Figure 1, due to no ATI, the regression lines for the dynamic copies model group and the static copies model group are parallel. Because of the significant difference of the type of instruction main effect, the dynamic copies model was significantly better than the static copies model in teaching recursive programming. Similarly, because of the significant difference of the prior knowledge main effect, high prior knowledge students outperformed low prior knowledge students no matter they were instructed with the dynamic or static copies mode.

### Table 4: Summary of Regression Analysis on the Type of Instruction and Prior Knowledge Main Effects for the Immediate Posttest

<table>
<thead>
<tr>
<th>Variable</th>
<th>B</th>
<th>SE B</th>
<th>β</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRP</td>
<td>-1.1359</td>
<td>.3947</td>
<td>-1.598*</td>
</tr>
<tr>
<td>PK</td>
<td>.7387</td>
<td>.0744</td>
<td>.5517*</td>
</tr>
<tr>
<td>(Constant)</td>
<td>11.7328</td>
<td>.7016</td>
<td></td>
</tr>
</tbody>
</table>

Note. $R^2 = .3093$, GRP = type of instruction, PK = prior knowledge. *$p < .05$.

![Figure 5. The nature of prior knowledge and type of instruction main effects.](image)

### Analysis of the Delayed Posttest

The delayed effect of the treatment was assessed by conducting the delayed posttest which measured participants' understanding of recursion two weeks after the treatment. The group means of the delayed posttest are listed in Table 5. The dynamic group had a larger group mean and smaller
standard deviation than the static group.

**Table 5** Group Means of the Delayed Posttest

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>SD</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>For Entire Population</td>
<td>11.45</td>
<td>4.07</td>
<td>230</td>
</tr>
<tr>
<td>Dynamic Group</td>
<td>11.88</td>
<td>3.71</td>
<td>114</td>
</tr>
<tr>
<td>Static Group</td>
<td>11.03</td>
<td>4.37</td>
<td>116</td>
</tr>
</tbody>
</table>

As shown in Table 6, the type of instruction × prior knowledge interaction was significant, $F(1, 229) = 14.3815, p = .0002$. The R-square was .3025, and the R-square change of the type of instruction × prior knowledge interaction was .0452. In other words, 4.52% of variance in the delayed posttest was accounted for by the type of instruction × prior knowledge interaction. Because the type of instruction × prior knowledge interaction was significant, multiple regression analyses were conducted on the delayed posttest for each treatment group in the following sections.

**Dynamic copies model group.** The prior knowledge main effect was significant, $F(1, 113) = 5.7923, p = .0177$. The R-square of the prior knowledge main effect was .0492, indicating that for the dynamic copies model group 4.92% of variance in the delayed posttest was accounted for by prior knowledge. The summary of regression analysis is shown in Table 7. The regression equation of the delayed posttest for the dynamic copies model group was:

$$Y_{\text{Dynamic}_{2}} = 10.1536 + 0.3347 \text{PK}$$

**Table 6** Summary of Regression Analysis on the Type of Instruction × Prior Knowledge Interaction for the Delayed Posttest

<table>
<thead>
<tr>
<th>Variable</th>
<th>B</th>
<th>SE B</th>
<th>β</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRP</td>
<td>-4.8118</td>
<td>1.3947</td>
<td>-.5924*</td>
</tr>
<tr>
<td>PK</td>
<td>-3.3234</td>
<td>.2944</td>
<td>-.2113</td>
</tr>
<tr>
<td>GRP × PK</td>
<td>.6681</td>
<td>.1762</td>
<td>.8877*</td>
</tr>
<tr>
<td>(Constant)</td>
<td>14.9654</td>
<td>1.6646</td>
<td></td>
</tr>
</tbody>
</table>

Note. $R^2 = .3205$, GRP = type of instruction, PK = prior knowledge. *p < .05.

**Table 7** Summary of Regression Analysis on Prior Knowledge for the Dynamic Copies Model Group on the Delayed Posttest

<table>
<thead>
<tr>
<th>Variable</th>
<th>B</th>
<th>SE B</th>
<th>β</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK</td>
<td>.3347</td>
<td>.1424</td>
<td>.2218*</td>
</tr>
<tr>
<td>(Constant)</td>
<td>10.1536</td>
<td>.7930</td>
<td></td>
</tr>
</tbody>
</table>

Note. $R^2 = .0492$, PK = prior knowledge. *p < .05.

**Static copies model group.** The prior knowledge main effect was significant for the static copies model group, $F(1, 115) = 91.5271, p < .0001$. The R-square of the prior knowledge main effect was .4453, indicating that for the dynamic group 44.53% of variance in the immediate posttest was accounted for by prior knowledge. The summary of regression analysis is shown in Table 8. The regression equation of the delayed posttest for the static copies model group was:

$$Y_{\text{Static}_{2}} = 5.3418 + 1.0128 \text{PK}$$

**Table 8** Summary of Regression Analysis on Prior Knowledge for the Static Copies Model Group on the Delayed Posttest

<table>
<thead>
<tr>
<th>Variable</th>
<th>B</th>
<th>SE B</th>
<th>β</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK</td>
<td>1.0128</td>
<td>.1059</td>
<td>.6673</td>
</tr>
<tr>
<td>(Constant)</td>
<td>5.3418</td>
<td>.6680</td>
<td></td>
</tr>
</tbody>
</table>

Note. $R^2 = .4453$, PK = prior knowledge. *p < .05.

As shown in Figure 2, ordinal ATI was found. High prior knowledge students performed better in the static copies model instruction than in the dynamic copies model instruction. In contrast, low prior knowledge students performed better in the dynamic copies model instruction than in the static copies model instruction.
Discussion

From the results of the analysis on immediate effects, the dynamic copies model which showed the dynamic process of recursion in animation was found to be more effective than the static copies model. The results were in accordance with the cognitive load theory. The dynamic copies model effectively reduced the cognitive load in learning recursive programming by providing more explanatory illustrations than did the static copies model.

![Delayed Posttest](image)

Figure 6. The nature of prior knowledge × type of instruction interaction.

The dynamic copies model, however, did not circumvent for students’ low prior knowledge. High prior knowledge students performed better than low prior knowledge students no matter they were instructed with the dynamic or static copies model. The reason probably due to the effectiveness of the static copies model which is conceived as a good model of recursion and is possessed by experts. The results of the current study were in accordance with Mayer and Sims’ (1994) finding which suggested that both high and low prior knowledge students benefited from good instruction. As found in previous studies (Aman, 1992; Clarke, 1992; Carlson & Wright, 1993), prior knowledge was a reliable predictor of achievement in learning computer programming.

For the delayed effects, ATI was found. High prior knowledge students benefited from the static copies model instruction more than from the dynamic copies model. In contrast, low prior knowledge students benefited from the dynamic copies model more than from the static copies model. The results were in accordance with previous research (Weinert, Helmke, & Schneider, 1989; Mayer, 1981) which suggested that the differences between individuals’ prior knowledge can be reduced by providing more instructional cognitive support.

In the view of instructional designer, instructional methods must go beyond simply providing information and support cognitive processing and reduce cognitive load when learners are incapable of doing so. From the results of the current study, cognitive load theory can be used as a useful guideline in designing effective instruction. The results of the current study also support that dynamic copies model is an effective instructional method for low prior knowledge students to learn recursion. High prior knowledge students, however, are suggested to allow more flexibility and alternatives in learning computer programming.

References


動態複製模型對遞回程式設計教學之影響

陳明溥
國立臺灣師範大學資訊教育學系

遞迴概念（recursion）之複製模型（copies model）以動態及靜態兩種形式被製作成電腦輔助教學軟體（computer-based instruction）在本研究中使用。在教授遞迴概念之立即效果方面，動態複製模型（dynamic copies model）比靜態複製模型（static copies model）更具有顯著效果。不論是使用動態或靜態複製模型，具有較高背景知識（prior knowledge）的學習者比具有較低背景知識的學習者有更好的學習效果。然而，在延遲效果方面，則出現了教學形式與背景知識間的交互作用（aptitude-treatment-interaction, ATI）。高背景知識的學習者在透過靜態複製模型學習時會有較好的學習效果，而低背景知識（prior knowledge）的學習者則在透過動態複製模型學習時會較好的效果。

關鍵詞：電腦輔助學習 概念模型 程式設計 認知負載