December 1998

The Use of Analogical Reasoning to Enhance Event-Driven Programs

Teresa Shaft
University of Tulsa

Follow this and additional works at: http://aisel.aisnet.org/amcis1998

Recommended Citation
http://aisel.aisnet.org/amcis1998/333

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 1998 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
The Use of Analogical Reasoning to Enhance Event-Driven Programs

Teresa M. Shaft
Department of QM and MIS
The University of Tulsa

Abstract

This study investigates the use of analogical reasoning while enhancing programs written in an event-driven language. As the field of information systems (IS) shifts from a single, dominant programming paradigm (procedural) to multi-paradigmatic (object-oriented, event-driven, and combinations) IS developers frequently find themselves working in new environments, frequently without formal training. The ability to transfer knowledge, i.e., use analogical reasoning, may allow developers to apply their skills in new environments more effectively. This study uses protocol analysis to investigate student developers use of analogy while enhancing a Visual Basic (VB) project.

Introduction

The field of IS has always been known for its rapid change, but the recent explosion of software development tools and shifts in programming paradigms makes keeping up-to-date nearly impossible. One ability that aids programmers working in new paradigms is to transfer knowledge, i.e., to apply knowledge learned in one context to a similar problem. Studies of transfer, or analogical reasoning, are commonly found in the cognitive psychology literature, but not in the IS literature. Hence, studies of transfer may yield insights of how IS developers can apply their current knowledge to new paradigms.

Theoretical Background

A computer-based literature search did not identify any studies of transfer between programming paradigms or program enhancement in non-procedural programming paradigms. Therefore, this study relies upon a theoretical model of program enhancement developed for procedural programming. Littman et al. (1986) found that programmers who used a systematic strategy to enhance a computer program were able to successfully complete an enhancement, while those that used an as-needed strategy were not successful. The current study investigates if this finding holds in an event-driven paradigm.

Proposition 1: The use of systematic strategy will be associated with higher scores on the experimental task.

Transfer is a well-known psychological phenomena (cf. Mayer, 1983). Near transfer is consistent with solving problems that are quite similar in structure to those with which the problem solver is familiar. Far transfer is consistent with solving problems that are less similar in structure. In this study, near transfer would be the application of knowledge learned within the event-driven paradigm. This study investigates novices’ ability to apply near transfer in the context of enhancing an existing project. Specifically, based on their knowledge of the new programming language (VB) participants should be able to complete the experimental task. The enhancement task in this study required a participant to add a “text box” and a “label” control to display data from a database. The original program contains two “text boxes” and “labels” which display data from the same database. Therefore, participants could rely on near transfer to complete the enhancement:

Proposition 2: The use of analogy is beneficial to completing the enhancement task; i.e., the more a developer makes use of analogy, the higher their score on the enhancement task.

Proposition 3: The higher the developer’s score on the relevant homework assignments, the higher their score on the enhancement task.

Procedures

The researcher gave a short introduction to VB in class after which students completed tutorial-based assignments (Gurewich and Gurewich, 1995). After completing the assignments, students individually completed the experimental task on-line. All students participated in the on-line task which was worth approximately 5% of their final grade. All students participated in the on-line task. Students from 3 semesters of the Systems Analysis and Design course participated. The eleven students enrolled the first semester served as pilot subjects, yielding eight usable protocols. The forty students from the final two semesters comprised the main study, with 35 usable protocols.

During the experimental session, participants enhanced a VB project as described above. Participants were asked to ‘think aloud’ as they worked on the enhancements. The audio signal and the signal from the computer monitor were sent to a videotape, providing a trace of each developer’s cognitive process. The video tapes were analyzed to: 1) identify the strategy used to complete the enhancements (as-needed or systematic), and 2) investigate the use of analogical reasoning during the enhancement process.
To examine Proposition 1, each participant’s strategy (systematic or as-needed) was identified, based on the definitions provided by Littman et al. (1986). All protocols were coded by the author and a research assistant who was not familiar with the research questions (raw agreement = .83, kappa = .43). Kappa did not reach conventional standards for agreement (≥.61). This was partially due to the high occurrence of the “as-needed” strategy resulting in estimate of chance agreement of .71. To address this issue, the coders met and resolved all coding disagreements prior to statistical analysis.

To examine the second proposition, a protocol coding scheme was developed from the pilot protocols. Three types of analogy were identified: textbook, control, and property analogies. A textbook analogy occurs when a participant states that they recognize the program as similar to one they completed as a homework or they refer to the textbook for general information. A control analogy occurs when the participant uses analogical reasoning to determine the appropriate control (for this task, both a label and text control are required). During protocol analysis, the names of the controls which are the source and target of the analogy are recorded. Controls have associated properties which define their characteristics such as width, color, font, etc. A property level analogy occurs when the participant uses analogical reasoning to determine the appropriate property and its value. The label and text controls would typically have their ‘name’, ‘width’, ‘height’, and ‘left’ properties altered. The text control must have the datasource and datafields properties set to ‘data1’ and ‘onhand’ respectively to complete the enhancement. As with control analogy, during analysis the names of the source and target controls are recorded. In addition, the name of the property and the value are recorded. For both control and property analogies, the original source of the analogy (textbook, the existing program, or an old program) is recorded. Finally, although no code is needed to implement the enhancement, participants attempted to make analogies based on code (a VB control may have code associated with it). Therefore, coders identified when participants attempted to draw analogies based on code in the existing program. Both successful and unsuccessful analogies are coded and accuracy is recorded. ‘Correct’ indicates the analogy “worked”. When an analogy failed, a reason (wrong control, wrong property, did not identify the correct property, not complete, did not identify the correct value, examined code, not needed or other) is recorded. Similar to the strategy coding, both coders’ worked independently and raw agreement = .90 and kappa = .74.

The third proposition is analyzed based on student’s scores on the task and relevant homework assignments. Grading criteria were developed and all tasks graded by the same individual.

Results and Discussion

Proposition 1. The use of a systematic strategy was not prevalent nor did it provide an advantage for completing the enhancement. Only a small percentage (5 of 35 or 14%) of participants used a systematic strategy. Further, use of a systematic strategy was not associated with higher levels of performance than observed among those who used an as-needed approach (chi-square = .96; p = .33). This result is inconsistent with Littman et al.’s (1986) finding that programmers who used a systematic strategy were successful, while those who used an as-needed study were not. There are several possible explanations for this difference. First, tools such as VB, which allow programmers to easily observe a system in action, may support an as-needed strategy better than procedural languages. Second, the experimental task was relatively straight forward, hence the advantages of a systematic strategy may not have been manifested. This explanation questions the generalizability of Littman et al.’s finding in that a systematic strategy may not always be advantageous. Those familiar with Littman et al.’s study may recall that the systematic strategy was necessary to understand that the delete function in their original application did not physically delete the record, but marked it as inactive causing it to be bypassed when records were displayed. Their experimental task to introduce an ‘undelete’ capability was more easily accomplished by programmers who understood details of the existing delete function. Clearly, more research is needed to differentiate between tool and task factors.

Proposition 2 investigates the effect of using analogy on performance, i.e., how well the participant completed the task. Recall that several types of analogies were coded. Due to space limitations only the most interesting results are reported. The correlation between the number of analogies (of any type) and performance was not significant (r = -.244, p = .16), nor that between the number of correct analogies and performance (r = -.074, p = .67). The only significant correlation between performance and measures of analogy was that between the number of incorrect control analogies and performance (r = -.395, p = .02). The more incorrect analogies a participant make concerning control, the worse their performance. This finding seems reasonable given the nature of the task, adding a textbox (control) and linking it to the Access database. The correlation between number of incorrect textbook analogies and performance reached close to conventional significance levels (r = -.28, p = .104). Similar to the finding relating to incorrect control analogies, the more incorrect textbook analogies (i.e., referred to the textbook, but based on the analogy on inappropriate information – such as a sequential file rather than an Access database) made by a participant, the lower performance. In general, making analogies did not improve performance, however drawing incorrect textbook or control analogies degraded performance.

Proposition 3 examines the relationship between performance on the VB assignments and the task. These analyses are based on a sample of 40, since this proposition is investigated based on the final saved project rather than protocols. The score on all VB assignments was positively related to performance on the task (r = .33, p = .04). Empirical evidence that doing ones homework pays! Surprisingly, the score on the specific assignment that the task was based on did not relate to performance (r = .07, p = .69). However, this may be a measurement artifact as the range for the specific task is much narrower than that for all VB assignments. Interestingly, there was a relationship between score on the VB homework and number of incorrect control
analogies \( r = -.37, p = .03 \). Recall that making incorrect control analogies was inversely related to performance \( p = .02 \). Possibly the link between performance and scores on the VB homework may be through the ability to recognize which control is needed. Note that a participant may place the correct control on the VB form without stating an analogy, hence the relationship is shown by the inverse relationship to number of incorrect analogies rather than a directly with correct analogies. Clearly, more sophisticated analyses are needed to confirm this possibility.

**Conclusion**

This study provides insight in the use transfer (analogy) in program enhancement. First, these findings are inconsistent with Littman et al.’s (1986) regarding the advantages of a systematic strategy. However, further research investigating other tasks and a more reliable coding process are needed. Second, a positive relationship between the use of analogical reasoning and performance was not detected, however programmers who made incorrect analogies had lower levels of performance. Hence, the inability to apply analogies degraded performance. It was surprising that so many participants had difficulty with the analogy given the ‘nearness’ of the transfer. Third, performance on the experimental task more strongly associated with performance on all VB assignments rather than the specific assignment on which the experimental task was based. Finally, those who scored highest on VB assignments had lower levels of incorrect control analogies. This inability to recognize the correct control upon which to based the analogy may be the link between score on the VB assignments and performance on the experiment task. This study begins the process of applying and updating theories of computer program enhancement developed from studies of procedural programming to other programming paradigms. It does not appear that theories developed under the procedural paradigm can be universally applied to other paradigms.

**References**

Grurewich N. & Grurewich, O. *Teach Yourself VB 4 in 21 Days*. Sam’s Publishing, Indianapolis, IN, 1995
