Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1995 Proceedings

Americas Conference on Information Systems (AMCIS)

8-25-1995

Application Domain Knowledge in Computer ProgramComprehension and Enhancement

Teresa M. Shaft
The University of Tulsa, shafttm@utulsa.edu

Application Domain Knowledge in Computer Program Comprehension and Enhancement

Teresa M. Shaft
The University of Tulsa
shafttm@utulsa.edu

Introduction

Software development transforms the conceptual models of the application domain into the formal models of the implementation (programming) domain (Blum 1989). The application domain is where the conceptual models, which describe what is to be done and possible approaches for a solution, are generated. These conceptual models account for many of the "essential difficulties" in software development (Brooks 1987). Research investigating software development, however, has focused on the implementation domain or computing element (Glass and Vessey 1992). Recent thinking indicates that investigations into the problems of software development must begin to consider the application domain (e.g., Guindon 1990).

Software maintenance is one task where application domain knowledge may play an important role. Maintenance remains important to organizations because maintenance costs, as a percentage of software expense, are expected to remain relatively constant (Foster 1991). Of maintenance activities, this research investigates enhancement because it accounts for the greatest percentage of person-hours expended on maintenance (Lientz and Soloway 1980, Abran and Nguyenkim 1991).

To enhance a computer program, programmers need to comprehend the existing program. It is estimated that maintenance programmers devote 50%-90% of their time to understanding the existing program, with the remaining 10%-50% spent implementing the enhancement (Robson et al. 1991).

The present research is an experimental investigation of the role of application domain knowledge in comprehension and enhancement. Establishing the importance of application domain knowledge in comprehension and enhancement may provide insight into why these tasks account for such a high percentage of software related costs.

Conceptual Model

The computer program comprehension and enhancement model used in this research is an elaboration of Letovsky's (1986). The comprehensive process results in the programmer formulating a mental model of the computer program, i.e., his or her understanding of it and how it operates. The programmer uses his or her existing knowledge base to understand a computer program. Two kinds of knowledge are of interest to the current research, application domain knowledge and programming knowledge.

The mental model needs to represent the numerous types of information depicted by the computer program (Pennington 1987, Curtis et. al 1989). Part of the essential difficulty of building and maintaining software comes from the difficulty of representing these multiple types of information (Brooks 1987). This research uses Pennington's (1987) approach (i.e., function, data flow, state, and control flow) because they have been used to investigate comprehension and enhancement.

To enhance a computer program, a programmer uses his or her mental model of the existing program, the program itself, and the enhancement specification. From this model, the following hypothesis were developed:

Hypothesis 1: A programmer will have higher levels of comprehension when studying a computer program from a familiar application domain.

Hypothesis 2: Enhancement tasks that affect a type of information closer to the application domain (e.g., function) will be easier to implement if the programmer has application domain knowledge. Enhancement tasks that affect a type of information farther from the application domain (e.g., control flow) will not be affected by the programmer's level of application domain knowledge.

Hypothesis 3: After conducting an enhancement task, programmers will have increased levels of comprehension with respect to the type of information affected by the enhancement task.

Methodology

To investigate the above hypotheses, 24 IS professionals, with an average of 10.7 years experience, studied and enhanced computer programs from two application domains, one familiar and one unfamiliar. Each programmer participated in a practice segment, then segments for each domain. Each segment consisted of: 1) a study period, after which programmers responded to a set of comprehension questions, and: 2) an enhancement period, during which programmers worked on an enhancement task and then responded to a second set of questions. Each question set contained 20 true/false questions, five for each type of information.

The order of presentation of the domains was counter-balanced, as were the two sets of equivalent comprehension questions.

COBOL was selected as the programming, language, accounting and hydrology as the familiar and unfamiliar application domains. The study required three programs, a practice program and one program for each domain. The accounting and hydrology programs were of equivalent complexity based on SLOC, 417 and 416 respectively. The DATA and PROCEDURE divisions were also comparable.

Enhancement tasks were necessary to test Hypothesis 2. For each domain, two enhancements were developed. Based on the definitions for the four types of information

(Pennington 1987), function and control flow tasks were selected as being the "closest" and "furthest" from the application domain. A single page specification described the required changes. The researcher implemented each enhancement to ensure that they resulted in programs of equivalent complexity.

Analysis and Results

Hypothesis 1. This hypothesis was investigated using programmers' responses to the first administration of comprehension questions in an ANOVA with two within-subjects factors (domain and type of question). Although no specific predictions concerning question type are specified for Hypothesis 1, question type was included since it is part of the overall experimental design. Error rates for the questions relevant to each type of information were the dependent variable. Programmers were expected to have lower error rates in the familiar domain and this hypothesis was supported ($p = .001$). Question type was also significant ($p = .001$). A follow-up analysis revealed that the error rate on data flow questions was significantly higher than state questions (51.25 v. 32.92, $p < .05$).

Hypothesis 2. Programmers were expected to achieve higher levels of performance on function tasks when working in the familiar domain and on control flow tasks in the unfamiliar domain. The hypothesis was tested using an ANOVA with one within-subjects factor (domain) and one between-subjects factor (task type). A participant worked on the same type of enhancement task in both domains. A score reflecting the enhancement quality was the dependent variable.

The anticipated interaction between task type and domain was not significant ($p = .65$), and the hypothesis was not supported. Domain was the only significant effect ($p = .03$); programmers' scores on the enhancement tasks were higher in the accounting domain than the hydrology domain (52.42 v. 38.96).

Hypothesis 3. Following the enhancement task, programmers were expected to have increased comprehension of the type of information affected by the enhancement task. The hypothesis was tested using an ANOVA with two within-subject factors (domain and question type) and one between-subjects factor (task type).

The dependent variable was the change in error rates on each type of comprehension question. Positive values indicate increased comprehension, negative values decreased comprehension.

The expected interaction between task and question type was not significant ($p = .31$). The effect of type of enhancement task ($p = .02$) was the only factor that was significant. Programmers who conducted control flow enhancements experienced greater increases in comprehension than those who conducted function enhancements (12.29% v. 2.92%). The domain by question type interaction was marginally significant ($p = .056$). A follow-up analysis of the interaction revealed that the only significant difference was related to the function question type (-4.12% v. 16.67, $p < .05$).

Discussion

As expected, programmers' achieved higher levels of comprehension in the familiar domain. When programmers bring more relevant knowledge (i.e., application domain) to a task, they achieve higher levels of performance.

Hypothesis 2 suggested that programmers' application domain knowledge influences their ability to perform different enhancement tasks. Type of enhancement task was not significant; however, the main effect for domain was significant. Application domain knowledge resulted in higher levels of performance regardless of the type of enhancement task, suggesting that application domain knowledge is an important factor in conducting software development tasks.

Hypothesis 3 examined how programmers' understanding of each program changed following the enhancement. The type of enhancement task was significant: programmers who conducted the control flow enhancement experienced greater increases in comprehension. To interpret this result it is useful to consider Weiser's (1982) theory of slicing, which argues that to debug a program, programmers identify the "slice" of the program related to the bug. Similarly, during enhancement, programmers identify the slice of the program related to the enhancement and build the enhancement onto that slice.

With respect to this study, control flow tasks required the addition of a control break. The new code is embedded in the existing control structure of the program, which creates a fairly large slice to build on. Working with a large slice likely forced programmers to increase their understanding of the original program. Conversely, function enhancements required programmers to add a new capability to a program, building onto a fairly small slice of the original program. Programmers, therefore, did not need to increase their understanding of the existing program. Hence, the programmers who conducted the control flow enhancement showed a greater increase in comprehension.

The other interesting result with respect to Hypothesis 3, the domain by question type interaction, was due to changes in the error rate for function questions. Programmers working in the familiar domain had slightly lower levels of comprehension after the enhancement. To understand this result, recall that programmers had relatively high levels of comprehension in the familiar domain following the study period. Therefore, when working in the familiar domain, programmers may have focused closely the new function and their understanding of the original program suffered.

Conclusion

Application domain knowledge had a significant influence on programmer's ability to comprehend and enhance computer programs. Further, this research addressed program comprehension and enhancement, tasks that are thought to reside more in the implementation domain than the application domain. Such tasks are not generally considered to require large amounts of application domain knowledge; programmers are

expected to rely upon their programming knowledge. However, this study demonstrates the importance of application domain knowledge to computer program comprehension and enhancement.

Bibliography

Abran, A. & H. Nguyenkim, "Analysis of Maintenance Work Categories Through Measurement", *Proceedings of the Conference on Software Maintenance 1991*, Sorrento, Italy, October, 1991, 104-113.

Blum, B., "Volume, Distance, and Productivity," *The Journal of Systems and Software*, 9 (1989), 217-226.

Brooks, F., "No Silver Bullet," *IEEE Computer*, April (1987), 10-19.

Curtis, B., S. Sheppard, E. Kruesi-Bailey, J. Bailey & D. Boehm Davis, "Experimental Evaluation of Software Documentation Formats," *The Journal of Systems and Software*, 8 (1989), 167-207.

Foster, J., "Program Lifetime: A Vital Statistic for Maintenance," *Proceedings of the Conference on Software Maintenance 1991* , Sorrento, Italy, October 1991, 98-103.

Glass, R. & I. Vessey, "Toward a Taxonomy of Software Application Domains: History," *Journal of Systems and Software*, 17, 2 (1992), 189-199.

Guindon, R., "Knowledge Exploited by Experts During Software System Design," *International Journal of Man-Machine Studies* , 33 (1990), 323-342.

Letovsky, S., "Cognitive Processes in Program Comprehension," in *Empirical Studies of Programmers: First Workshop*, E. S. Soloway & S. Iyengar (Eds.), Ablex Publishing, Norwood, NJ, 1986, 58-79.

Lientz, B., E. Swanson & G. Tompkins, "Characteristics of Application Software Maintenance," *Communications of the ACM*, 21, 6 (1978), 466-471.

Pennington, N., "Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs," *Cognitive Psychology*, 19 (1987), 295-341.

Robson, D., K. Bennett, B. Cornelius & M. Munro, "Approaches to Program Comprehension," *The Journal of Systems and Software*, 14 (1991) 79-84.

Weiser, M. "Programmers Use Slices When Debugging," *Communications of the ACM*, 25, 7 (1984), 352-357.