8-15-1997

Java as a First Programming Language: A Meta-Analytic Review

Christopher G. Jones
Brigham Young University-Hawaii

Thomas F. Chan
Brigham Young University-Hawaii

Java as a First Programming Language: A Meta-Analytic Review

Christopher G. Jones
F. Thomas Chan
Brigham Young University-Hawaii

Introduction

A recent market study by Forrester Research, Inc. proclaims "Java will become the universal language of Internet computing" (Stewart, 1997, p. 12). International Data Corp predicts that the number of Java programmers will quadruple from 200,000 to 800,000 by the year 2000 (Stewart). Not only is interest in Java accelerating in the corporate world, interest continues to grow in academia. According to Sun Corporation, creator of the Java language, over 100 academic institutions throughout the world have already integrated Java into their curriculum (Lovell, 1997). Within the past year, several educators have proposed the adoption of Java as the first language for instruction in programming (Lea, 1996; Wallace, C., Martin, P. & Lang, B., 1997). Yet research in programming language instruction suggests adopting a new programming language is not a curriculum decision to be taken lightly (Brilliant & Wiseman, 1996). Recently, some experience reports regarding the introduction of Java into the curriculum have been made available. And early this year, entire conferences (e.g., Java in the Computing Curriculum) and conference panels (e.g., Java in the C.S. Curriculum, SIGSCE '97) were devoted to the topic. To date, however, little systematic research has been done to synthesize the reported findings. To address this need, a meta-analysis was undertaken using a modification of the quantitative approach popularized by Glass (1976).

Methods and Procedures

The traditional quantitative meta-analysis requires the researcher to (a) identify a representative sample of studies through objective and replicable searches; (b) code the studies for salient characteristics, quantifying each study outcome on a common metric; (c) using statistical analysis, systematically exam how study outcomes covary with salient study characteristics; and (d) develop conclusions based on the findings from analysis that are explicit and replicable (White, 1993). Given that much of the available study data consists largely of anecdotal experience reports with little attention to systematic measures of student outcomes, a decision was made to forgo the search for a common metric and an analysis of covariance. Instead, each study was coded by the authors for purported benefits arising from the use of Java as a teaching language, concerns developing over such use, and recommendations for the adoption of Java. Coded studies were tabulated into frequency counts. The results are presented in Tables 2 - 4.

The studies considered for inclusion in this meta-analysis were identified from four sources: (a) on-line searches of the world wide web using key words and key word combinations for such terms as Java, computer science, information systems, programming, education, and teaching; (b) a review of publications targeted to object technologists, and computer science and information systems educators; (c) bibliographic references cited in the studies located through computer searches and periodical reviews; and, (d) a review of listservs dedicated to computer science and information systems education.

A total of 18 articles related to programming education and Java were located through these search procedures. This candidate pool of articles was reduced to seven studies (Table 1) for inclusion in the meta-analysis based on the following criteria: (a) studies had to explore the merits of Java as a first programming language; (b) studies had to be targeted to an educational setting, with no restrictions on grade level or delivery institution; and, (c) due to cost and time considerations, studies had to be published and be available from university libraries or be readily available over the Internet.

Since education and training related to Java is just in its infancy, few of the studies showed any methodological rigor. As mentioned earlier, most of the included studies were experiential in nature, primarily providing anecdotal narrative about attempts to design and deliver an introductory course in Java. Study narratives did not provide any evidence of random selection of students, nor use of control groups

with a corresponding random assignment of students to control and experimental groups, nor use of any instrumentation.

While effect size (ES) is used as a common metric in traditional meta-analyses, sufficient statistical data were not provided in the seven studies to develop such a standardized outcome measure. Of these seven studies, only three reported any outcomes. Wallace (1997) reported increased language apprehension over Smalltalk or C++; Lea (1996) reported decreased student attrition over previous courses taught using C++ as the first language; and Clark (1996) reported that students found Java difficult to learn. Although none of these metrics were used to assess directly the appropriateness of Java as a teaching language, they do serve as rough surrogates for an indirect approximation of the impact on student learning. Perhaps the most useful data collected were frequency data on (a) claimed benefits of using Java as a pedagogical language, (b) issues arising from the introduction of Java into the classroom, and (c) recommendations regarding the use of the Java as a teaching language. For each study, benefits, issues, and recommendations were listed and categorized.

Findings

The meta-analysis of the claimed benefits, issues, and recommendations concerning the use of Java are presented in Tables 2 - 4. Purported benefits have been categorized into those related to the impact of Java on object technology migration, those facilitating student learning, and those providing superior language constructs over other first languages. Issues have been categorized into two groups: (a) issues concerning the current Java language and development environment, and (b) issues concerning student learning. Finally, recommendations have been grouped into advice regarding instructional approach, advice regarding course content, and advice to those considering adoption of Java.

Discussion

The results of this meta-analysis indicate that for those reporting some form of outcome, introduction of Java appears to have a mixed impact on student learning and attitude; two studies claimed some positive benefits from using Java, while one study indicated a detrimental impact on student learning. The most useful data to be gleaned from the analysis are the summaries of benefits, issues, and recommendations (Tables 2 - 4). Based on these summaries, the primary reason given for migrating to Java is that by eliminating pointers, adding garbage collection, and proving a fully dynamic model for object creation, Java is in many respects a better C++. With Java's class libraries and full-object orientation, it encourages reuse and compositional software construction more so than C++. Some of the studies indicated that educators were able to address more advanced material in a Java version of an introductory programming course than they normally were able to in C++ version of the same course.

The issues summary (Table 3) indicates that lack of adequate curricular materials is the single biggest impediment to successful Java instruction. Another major concern was unreasonable student expectations regarding how easy it is to program in Java programming. A few of the studies indicated that the Java's mix of objects and native types confused students. Insufficient support for simple input/out made student programming more complicated than necessary. With regards to the language itself, educators accustomed to Ada generics or C++ templates found the lack of parameterized classes made object collections more problematic. Other language concerns included the volatility of the language standard and the inability to cleanly separate implementation from interface.

Recommendations (Table 4) were not nearly as frequently cited as benefits (Table 2) or language concerns (Table 3). Two studies recommended sequencing coverage of the object-oriented features of Java after coverage of language basics and procedural abstraction. Two studies recommended delaying the decision to adopt Java until better instructional materials were available or until the language matured. Finally, two study recommendations dealt with course content; one study recommended the adoption of Java or a Java-like language to facilitate the teaching of modern programming, and another suggested that traditional

computer science topics be integrated rather than taught in isolation so that design and engineering principles could receive additional attention.

All of these meta-analytic findings must be tempered by two serious threats to the validity of this study-the lack of methodological rigor in the original studies and the small sample size of studies for review. None of the studies analyzed referred to the use of random sampling, controls, or blind assessment. In fact all data is self-reported by those teaching or anticipating teaching Java courses. Further the small sample of seven studies calls into question the stability of the findings, especially where studies did not provide complete data regarding outcomes.

Conclusion

There is a danger in this meta-analysis of falling prey to Bangert-Drown's (1986) criticism of "garbage in-garbage out" meta-analyses. Surely there is a glaring need for rigorous research to determine the appropriateness of Java as the first language of instruction. However, even the non-scientific studies that comprise this review do provide valuable insights into issues surrounding the introduction of Java into the curriculum. Aggregation of previous unaggregated research gives some feel for the direction of the data. These findings, though based on anecdotal evidence, provide a starting point for developing hypotheses regarding a possible migration to Java. At Brigham Young University-Hawaii efforts are already underway to explore the use of Java in the introductory programming course. Though it is still too early to tell, early results in the classroom tend to confirm the findings of this study. Students enjoy programming in Java, find it easier than C++, but harder than a traditional procedural language. Additional research is needed and is in progress.

References and Tables 3-4 available from the first author.

Email: jonesc@cs.byuh.edu

Table 1
Studies Reviewed, Coded, and Tabulated

Study No.	Author(s)	Title
1	Wallace, C., Martin, P. & Lang, B.	Not *whether* Java but *how* Java (italics in original)
2	Lea, D.	Some Questions and Answers about using Java in Computer Science Curricula.
3	Hosch, F.	Java as a First Language: an Evaluation.
4	Skrien, D.	Java in the first year.
5	Stoecklin, S.	Java in CS1; pedagogically sound. OOPSLA panel.
6	Clark, D.	Java is easy.
7	Bergin, J.	Java as a Better C++.

Table 2

Purported Benefits of Using Java as Pedagogical Language

Purported Benefit	Frequency	%
Migration to Object Technology		
Eases understanding of OO concepts through adoption of a fully dynamic model for objects (1, 3, 7)	3	5.66
Encourages code reuse (5, 7)	2	3.77
Facilities introduction of software models of real world things through AWT (2, 7)	2	3.77
Provides conceptual distinction between type and class, via Interfaces (1, 7)	2	3.77
Fosters the introduction of objects early (2, 7)	2	3.77
Other migration to object technology benefits (1, 1, 2, 3, 6, 7)	6	11.32
Student Learning		
Eliminates the most problematic C++ syntax, such as pointers (1, 6, 7)	3	5.66
Manages dynamic memory automatically using garbage collection (1, 3, 6)	3	5.66
Simplifies student learning as the language is small and easier to learn than Ada (3) or C++ (2, 5)	3	5.66
Emphasizes mainstream programming through strong typing (3, 7)	2	3.77
Minimizes time to learn programming basics, facilitating introduction of advanced material (2, 7)	2	3.77
Provides gentle introduction to C/C++ (7, 2)	2	3.77
Other student learning benefits (1, 1, 2, 2, 2, 2, 7, 7)	8	15.08
Superior Language Features of Java		
Supports concurrency and distributed systems development (1, 2, 7)	3	5.66
Elevates modularization to the granularity level of a Package rather than that of a file or class (1, 7)	2	3.77
Tightly integrates error handling (1, 7)	2	3.77
Other language feature benefits (1, 5, 6, 7, 7, 7)	6	11.37
Total	53	100.0%

Note. References to study numbers are in parentheses.