Human Factors and Software Reuse: The Manager's Impact

John A. Lewis, Sallie M. Henry,
Dennis G. Kafura, and Robert S. Schulman

TR 92-13

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

April 13, 1992

Human Factors and Software Reuse: the Manager's Impact

by

John A. Lewis Sallie M. Henry Dennis G. Kafura

(Department of Computer Science)

and

Robert S. Schulman

(Department of Statistics)

Virginia Tech
Blacksburg, Virginia 24061

Internet: lewis@vtopus.cs.vt.edu

ABSTRACT

While some of software engineering's basic assumptions seem intuitive, the need for scientific experimentation remains clear. Empirical validation is the hallmark of a mature scientific or engineering discipline. However, little precise experimentation is performed in computer science, especially in the area of software engineering. Several assumptions are made about the factors affecting software reuse, specifically concerning the role of human factors such as managerial influence. This paper describes the results of a controlled experiment designed to evaluate the impact of human factors on software reuse. The experiment concludes that (1) software reuse promotes higher productivity than no reuse, (2) reuse resulting from both moderate and strong encouragement promote higher productivity than no reuse, and (3) while strong managerial encouragement did not create a significant difference in productivity, it does tend to promote improper reuse activities.

Keywords: software reuse, human factors, empirical studies

1. Introduction

The fundamental basis of any scientific discipline is precise, controlled empirical investigation. However, claims made by software engineers often remain unsubstantiated because they are inherently difficult to validate or because their intuitive appeal seems to dismiss the need for scientific confirmation.

Software reusability is one area where unverified claims exist. The potential for reusability to improve the software development process is immense, yet a variety of factors affect its impact. Several theories exist about factors which either help or hinder the reuse process, but for the most part these theories have not been verified.

Developers and users of potentially reusable products are unnecessarily hampered because they lack specific knowledge concerning the factors which influence software reusability. These factors affect whether existing components can be:

- evaluated for potential reuse,
- adapted to the required situation, and
- integrated into the new product.

Furthermore, the cost of the reuse process must stay below the cost of the development process or the main purpose of reuse is defeated.

These influential factors are both technical and non-technical in nature. Technical factors include development paradigm issues and tools for searching and classifying reusable components. The majority of work performed in the area of software reuse has focussed on these technical factors while non-technical issues, particularly human factors, are often ignored or dismissed. Tracz specifically addresses the myth that software reuse is only a technical problem [TRAW88].

The research described in this paper investigates some claims made about the factors which affect software reusability. A controlled experiment was designed and executed to evaluate the impact of managerial on software reuse. Another related issue, the object-oriented paradigm, was also investigated. The results of this aspect of the research

are discussed in [LEWJ91]. The scope of this paper focuses only on the managerial factors in question.

Despite any high technology used to improve state-of-the-practice reuse, it is the human mind which develops reusable products and decides if existing code should be reused. The study of human factors is a major psychological paradigm affecting the manner in which code is developed [CURB87]. Even though human qualities are often difficult to objectively quantify, they play a major a role and are therefore a primary issue in this research.

In a realistic software development environment, a majority of the work performed and the techniques by which that work is accomplished are greatly influenced by the production manager. Management's attitude is consistently mentioned in current literature as an important factor governing the successful reuse process [BIGT87] [TRAW88].

Managerial influence is an external stimulus which affects the software developer in a variety of ways. If the manager's attitude is apathetic or weak toward reuse, then it is left to the individual programmers to decide what emphasis to give to reuse. If, however, management's attitude encourages the reuse process, programmers might put more effort into correctly performing reuse tasks. Freeman makes the point that managers and technical people need different forms of reuse training [FREP83]. However, managerial influence raises another question. Can too much emphasis be placed on reuse? If programmers are pressured to reuse, inappropriate code might be integrated into the new system causing modification delays that otherwise would have been avoided.

The goal of the experiment described in this paper is to answer the following questions concerning the human factors affecting software reuse:

1) Does reuse promote higher productivity than no reuse?

2) Does management's moderate encouragement to reuse promote higher productivity than no reuse?

3) Does management's strong encouragement to reuse promote higher productivity than no reuse?

4) Does management's moderate encouragement to reuse promote higher productivity than management's strong encouragement to reuse?

5) Does management's strong encouragement to reuse promote improper reuse activities?

The experimental design was constructed with these questions in mind. We define productivity as the inverse of the effort expended to produce a specific software product. Effort is measured in several quantifiable ways.

The next section describes the design of the experiment and discusses the specifications of the tasks performed. Section 3 defines the data collected and the statistical analysis performed. Section 4 draws conclusions from the analysis, specifically addressing the questions presented above. Finally, Section 5 summarizes the experimental results and discusses future work in this area.

2. Experimental Design

Some reuse experiments employ hypothetical, question-and-answer situations where the subjects do not actually perform the various tasks inherent in the reuse process. We believe, however, that to accurately determine influential factors, the experimental subjects must perform all of the following tasks: evaluate potentially reusable products, adapt them to the new situation, and integrate them into a functionally complete product. It is important to create, as accurately as possible, a representative situation while maintaining a valid experimental design [CURB80].

The subjects in the experiment are a set of senior-level software engineering students. The use of students as subjects, while sometimes considered unrealistic, is justified in this case due to two overriding considerations. First, empirical evidence by Boehm-Davis indicates that advanced students are equal to professionals in many

quantifiable measures, including their approach to developing software [BOED84]. Although new techniques are learned and further refinement does occur, a programmer's basic approach and development habits are formed quite early in their professional development. Second, given the amount of control necessary to execute this experiment, students are the only viable alternative. The efficacy of students as subjects is supported for within-subject experiments by Brooks [BROR80], and therefore applicable to this research.

The subjects in the experiment developed a well-defined target system. The system specification is couched in the guise of computerizing a fictional company and is separated into two tasks. The specific functional operations making up the system were abstracted from commercial software development experience. Both tasks involve a variety of programming techniques including data management, numerical processing, and graphics. Reusable code components were made available to the subjects implementing the target system. To affect further control, the code component sets were specifically generated for this study, as described in the next section.

Previous research investigating the factors affecting software reuse has concentrated on two issues: (1) the effect of software engineering characteristics of code components, such as readability, structured code, etc., and (2) the techniques used to find appropriate code components from a set of possible candidates. Neither of these issues is the focus of this study. Code quality was allowed to vary only within the controlled confines of "adequate" testing and software engineering standards. All completed projects were verified to meet a set of requirements concerning documentation, code quality, and functional correctness. Furthermore, subjects were given no special tools for searching or identifying candidate components. It is assumed that any assistance in this area would only improve the reuse results. This study focuses on project managements influence on the reuse process.

The research consists of two phases. The first phase was preparatory, in which potentially reusable components were designed and implemented. The experiment was executed in the second phase, in which the target system was developed by a set of subjects, who are unrelated to the programmers who designed and implemented the reusable components. These two phases are described in the following sections.

2.1 Phase One: Component Development

Two sets of potentially reusable components were created during phase one. To ensure the experiment is not biased to a particular language paradigm, one set of components was written using the procedural paradigm (Pascal) and the other set was implemented using the object-oriented paradigm (C++). Both component sets were designed to be functionally equivalent, though design and coding techniques naturally vary between the two language paradigms. Equivalence was guaranteed by ensuring that all code meet the same fundamental functional and error-handling requirements. All developed code was required to pass the same level of testing thoroughness.

Knowing the requirements of the target system to be implemented in the second phase, each component was designed to have a specific level of applicability. The levels of reuse can be described as:

1) completely reusable,
2) reusable with "small" changes (< 25%),
3) reusable with "large" changes (> 25%), and
4) not applicable for reuse.

With respect to the target system, the component sets were designed to contain elements from each reuse level. The 25% delimiters of levels 2 and 3 are only subjective guidelines and refer to the amount of code that must be added, deleted and modified to create a component that meets the target system's requirements. Providing components which span a wide range of applicability ensures a realistic, verbose domain from which subjects evaluate and choose components.

2.2 Phase Two: Project Implementation

Using the two sets of components, independent subjects were assigned the task of implementing the target project. The subjects were divided into three groups, pictured as

cells in Figure 1. One group could not reuse at all, therefore implementing the project using only newly developed code. A second group was encouraged to reuse components from the appropriate reuse set as they saw fit. The third group was instructed that they should reuse anything remotely appropriate in the reuse set.

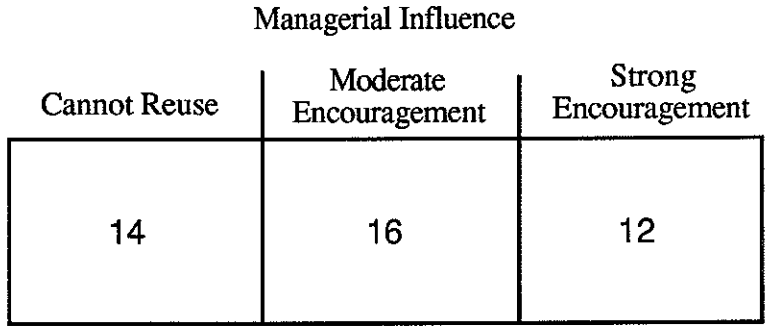

Figure 1: Group Breakdown with Number of Observations.

Twenty-one subjects were divided into the groups randomly, but were statistically blocked across their computer science grade point averages. Two observations per subject were collected resulting in a total of forty-two observations. The blocking was accomplished by dividing the students by grade point average into high and low groups, then randomly assigning subjects to blocks from alternating levels. This blocking was an effort to reduce variability within each group. An anova test comparing the grade point averages of subjects showed no significant differences between groups ($p < 0.9795$).

The functional requirements of the target system were divided into two equal tasks related to "employee management" and "business management." Employee management deals with an employee database, payroll, security control, and cost center management. Business management is concerned with the details of shop floor control, quality control testing, warehouse management, and customer interactions.

Although the two tasks focus on different aspects of running a business, they were designed to be comparable in programming difficulty. Both tasks are divided into seven subtasks, each of which has a counterpart in the other task designed to require

approximately the same amount of effort to develop. Initial analysis of the task factor determined that the difference between the two tasks played no role in influencing any of the productivity variables (all p-values for task effects were ≥ 0.2073). In other words, the two tasks were determined to be equally difficult. The lack of difference is attributed to the careful design of task specifications and the blocking of subjects across grade point average. Therefore, all further analyses ignore the task factor, effectively creating 42 observations on which to perform the tests, which gives them more statistical power.

To further control for task differences, half of the subjects designed and implemented the employee management task first, while the other half of the subjects designed and implemented the business management task first. Then each half switched, resulting in both system tasks being developed by each subject. This organization offsets any learning benefit of doing a particular task first.

3. Data Analysis

Given a feasible and well-controlled experimental design, the rigorous analysis of data collected during the experiment determines the conclusions that can be made about the hypotheses. In this experiment, the goal is to determine which groups from Figure 1, on average, had a significantly different productivity rate than others.

The data collected during the experiment measures the productivity of a subject during the implementation of the target system. Productivity and effort are considered to have an inverse relationship. Therefore, the less effort expended by a subject to satisfy the requirements of one task, the higher the productivity of that subject. The measurements of effort, and therefore of productivity, are:

Main productivity measures

- **Runs** - The number of runs made during system development and testing,
- **RTE** - The number of run time errors discovered during system development and testing,
- **Time** - The time (in minutes) to fix all run time errors,

Secondary productivity measures

- **Edits** - The number of edits performed during system development and testing, and
- **Syn.** - The number of syntax errors made during system development and testing.

Since each subject implemented the same tasks, a comparison of data across subjects yields a relative measure of the effort used to develop a system task. A subject with a high value for a given indicator is considered less productive than a subject with a low value.

Multiple productivity variables are used to obtain a complete picture of the development process. The Runs, RTE and Time variables, given their significance to the development process, are considered the main variables of interest. The Edits and Syntax Errors variables are gathered for completeness, but are given less emphasis. In an effort to reduce the overhead of the data collection, some measures that might have been of interest, such as total development time, were not collected.

Data was collected by the subjects using tally sheets. To assure the data's validity, subjects were informed that their names would not be associated with these data, and that the values themselves would have no bearing on their course evaluation. They were also told that a negative impact on their course evaluation would occur if they did not record their development information honestly and completely. The tally sheets were coded such that only a subject identification number was ever connected to particular data.

The group means for each productivity variable are depicted graphically in Figures 2 and 3. These charts give a rough indication of how the groups compare although statistical analyses are employed to verify the perceived differences. In each analysis, a difference in means was considered significant if the p-value for the test was less than 5% ($p < 0.05$), which is an accepted norm. Since our research questions all predict the direction of difference, all tests were performed in a one-sided manner.

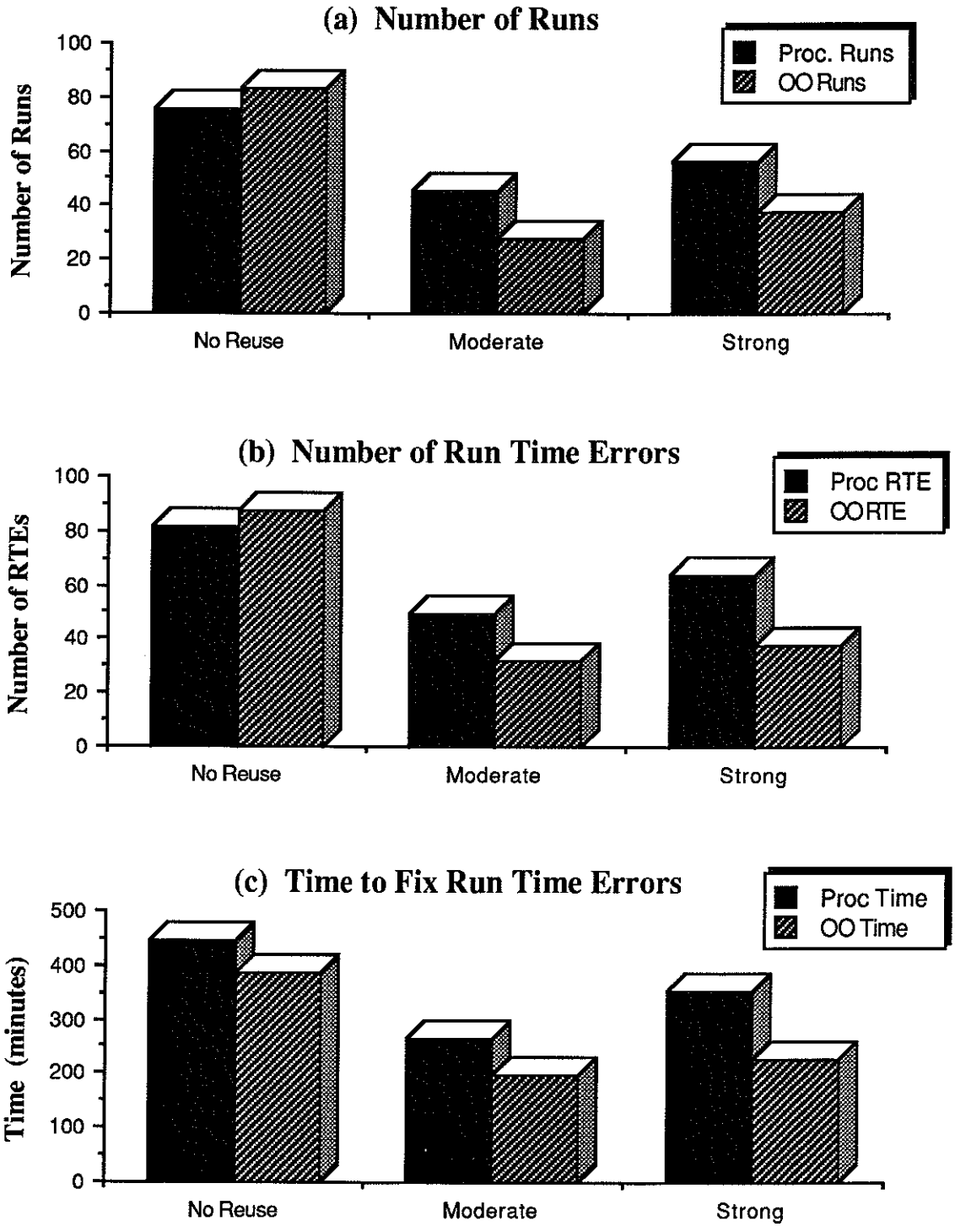

Figure 2. Group means for primary productivity variables.

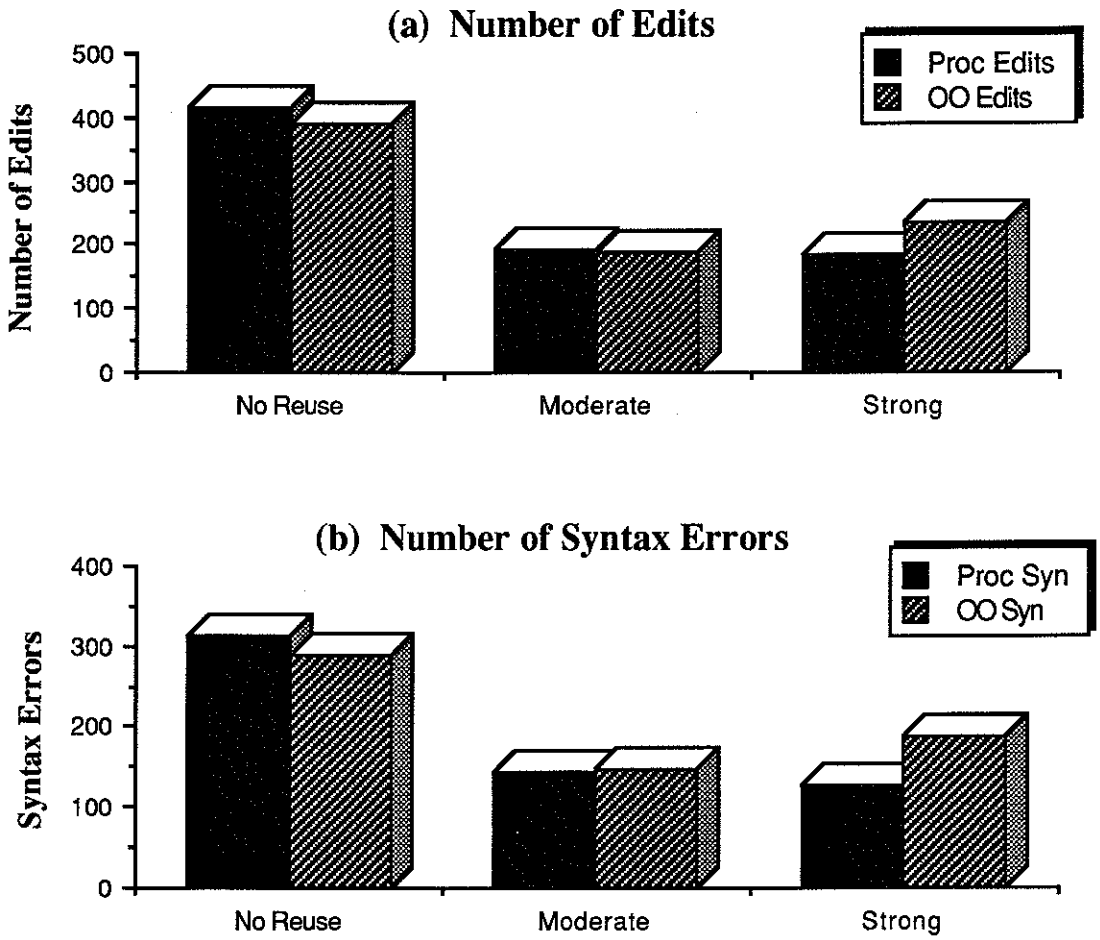

Figure 3. Group means for secondary productivity variables.

4. Empirical Results

For questions 1 through 4 (from section 1), tables are given which summarize the analyses performed. Question 5 deals with issues that do not lend themselves to straightforward statistical analysis given the available data, and is therefore discussed in a less rigorous manner.

Our hypotheses always support the reuse process. That is, assuming fundamental reuse facilities, any level of encouragement to reuse will result in higher productivity than no reuse at all. Furthermore, our hypotheses support moderate encouragement above strong encouragement, given the theory that overzealous reuse can lead to unproductive activities in modifying components that are not applicable to the target system.

The experimental questions posed in Section 1 are used as a framework for discussion of the statistical analysis. Each question is addressed separately, giving the results of the appropriate analysis.

1) Does reuse promote higher productivity than no reuse?

Given the results in Table 1, the answer to this question is clearly yes. The means in the third column of Table 1 are calculated for all subjects who did not reuse. Likewise, the fourth column shows means for all subjects who did reuse, combining the moderate and strong levels of managerial influence.

Table 1. No Reuse vs. All Reuse

			Means	
	Significant?	p-value	No Reuse	All Reuse
Runs	Yes	0.0001	78.71	41.14
RTE	Yes	0.0001	83.79	45.04
Time	Yes	0.0001	420.07	255.36
Edits	Yes	0.0001	405.71	198.82
Syn.	Yes	0.0001	302.14	150.92

Our hypothesis is that the means will be lower for the reuse groups, indicating a higher productivity for the subjects who were encouraged to reuse. This hypothesis is strongly supported by all variables.

2) Does management's moderate encouragement to reuse promote higher productivity than no reuse?

The means listed in the third column of Table 2 represent the moderate encouragement level of managerial influence. These means are compared to the "no reuse" group, given in column four. Our hypothesis is that the means of the third column will be lower than the means of the fourth column, representing a higher productivity rate for those subjects who were moderately encouraged to reuse.

Table 2. Moderate Encouragement vs. No Reuse

	Significant?	p-value	Means	
			Moderate	No Reuse
Runs	Yes	0.0001	36.44	78.71
RTE	Yes	0.0001	40.75	83.79
Time	Yes	0.0001	230.19	420.07
Edits	Yes	0.0001	190.81	405.71
Syn.	Yes	0.0001	145.00	302.14

All five productivity variables supported the hypothesis to a statistically significant degree. We conclude, therefore, that moderate encouragement to reuse promotes higher productivity than no reuse.

3) Does management's strong encouragement to reuse promote higher productivity than no reuse?

This question compares the group with strong managerial encouragement to reuse, represented by the third column of Table 3, to the group that did not reuse, whose productivity means are listed in the fourth column. Again favoring reuse over no reuse, our hypothesis is that the strong encouragement means in the third column will be less than the "no reuse" means in the fourth column.

Table 3. Strong Encouragement vs. No Reuse

			Means	
	Significant?	p-value	Strong	No Reuse
Runs	Yes	0.0001	47.41	78.71
RTE	Yes	0.0001	50.75	83.79
Time	Yes	0.0037	288.92	420.07
Edits	Yes	0.0001	209.50	405.71
Syn.	Yes	0.0001	158.83	302.14

Once again, all productivity variables support the hypothesis, with all means showing a higher productivity for the groups under strong managerial influence to reuse than the groups that did not reuse.

4) Does management's moderate encouragement to reuse promote higher productivity than management's strong encouragement to reuse?

Table 4 compares the means of the moderate encouragement group to those of the strong encouragement group. Because of the possibility of lost time and effort resulting from subject's attempts to reuse components which are not appropriate to the target system, our hypothesis is that the group with moderate encouragement will have higher productivity than the group with strong encouragement. That is, our hypothesis is that the means in the third column are less than the means of the fourth column.

Table 4. Moderate vs. Strong Encouragement

	Significant?	p-value	Means: Moderate	Means: Strong
Runs	Yes	0.0086	36.44	47.41
RTE	No	0.0556	40.75	50.75
Time	No	0.0910	230.19	288.92
Edits	No	0.2144	190.81	209.50
Syn.	No	0.1863	145.00	158.83

The means for the Runs primary variable supported the hypothesis to a significant degree. The p-values for the RTE and Time primary variables were not quite significant, although the means differed in the hypothesized direction. The secondary variables, while differing in the hypothesized direction, were not significant.

Since the difference in means was significant for only one primary variable, we cannot conclude that moderate managerial encouragement promotes higher productivity than strong encouragement. However, the other two primary variables barely missed the 5% significance level, and all five productivity indicators differed in the hypothesized direction. Therefore, the effect of managerial influence certainly warrants further investigation.

5) Does management's strong encouragement to reuse promote improper reuse activities?

The analysis of question 4, while unable to conclude categorically that moderate encouragement promotes higher productivity than strong encouragement, indicates a tendency for less productive results in the strong encouragement category. Question 5 is posed to determine if the cause for lower productivity is inappropriate reuse, as hypothesized.

An analysis of the modules most commonly reused by the moderate and strong groups gives an indication that this hypothesis has merit. The moderate group contained eight subjects and the strong group contained six subjects. Seven modules were consistently reused by the subjects in the moderate group, and that collection is called the "base set" hereafter. Six subjects used all seven modules, one used five of the seven, and one used only one. The maximum total modules reused by any subject in the moderate group was nine, while the minimum was one. Six distinct modules other than those in the base set of seven were reused, but with little consistency between subjects.

Compare this to the reuse practices in the strong encouragement group. Four subjects used the complete base set, one used six of the seven modules, and one used five of seven. However, the subjects in the strong group reused fifteen distinct modules other than those in the base set. The maximum number of modules reused by subjects in this group was fourteen, while the minimum was six. Once again, little consistency was shown in reusing the modules outside of the base set.

The average percentage of code reused from the base set of modules was 85% by the moderate group, and 81% by the strong group. The average percentage of code reused from all extra modules in the moderate group was 72%, compared to 55% for the strong group.

From question 4, the subjects in the strong group tended to have worse productivity than the moderate group, although not significantly so. Furthermore, there were certain consistencies in the reuse habits of all subjects, and the strong group reused modules that yield less actual code for the target system. Therefore, there is an indication that a cause-effect relationship exists, specifically that the strong encouragement to reuse caused subjects to reuse modules which were more productively written without the assistance of reusable components.

5. Summary

This paper describes an experiment investigating human factors affecting software reuse. The conclusions formed by this research are summarized below:

(1) Software reuse promotes higher productivity than no reuse (question 1),

(2) Reuse resulting from both moderate and strong encouragement promote higher productivity than no reuse (questions 2 and 3), and

(3) While a strong managerial influence did not create a significant difference in productivity, it does tend to promote improper reuse activities (question 5)

Although we did not demonstrate that management's moderate encouragement to reuse promotes higher productivity than strong encouragement, the results tend to indicate that this phenomenon exists to some degree. Further research in this area is strongly motivated by this analysis.

References

[BIGT87] Biggerstaff, T., Richter, C., "Reusability Framework, Assessment, and Directions," IEEE Software, March 1987, pp. 41-49.

[BOED84] Boehm-Davis, D., Ross, L., "Approaches to Structuring the Software Development Process," International Journal of Man-Machine Systems, (to appear 1991).

[BROR80] Brooks, R., "Studying Programmer Behavior Experimentally: The Problems of Proper Methodology," Communications of the ACM, 1980, Volume 23, Number 4, pp. 207-213.

[CURB80] Curtis, B., "Measurement and Experimentation in Software Engineering," Proceedings of the IEEE, 1980, Volume 68, Number 9, pp. 1144-1157.

[CURB87] Curtis, B., "Fifteen Years of Psychology in Software Engineering: Individual Differences and Cognitive Science," Proceedings of the Seventh International Conference on Software Engineering, March 1984, pp. 97-106.

[EKSR76] Ekstrom, R.B., French, J.W., Harmon, H.H., "Manual for Kit of Factor-Referenced Cognitive Tests," Educational Testing Service, August 1976.

[FREP83] Freeman, P., "Reusable Software Engineering: Concepts and Research Directions," ITT Proceedings of the Workshop on Reusability in Programming, 1983, pp. 2-16.

[LEWJ91] Lewis, J., Henry, S., Kafura, D., Schulman, R., "An Empirical Study of the Object-Oriented Paradgim and Software Reuse," OOPSLA '91, October 1991, (to appear).

[PRIR87a] Prieto-Diaz, R., Freeman, P., "Classifying Software for Reusability," IEEE Software, January 1987, pp. 6-16.

[TRAW88] Tracz, W., "Software Reuse Myths," ACM SIGSOFT Software Engineering Notes, January 1988, pp. 17-21.