# Management Issues and Software Reuse: An Empirical Study

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

TR 92-16

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

April 13, 1992

# Management Issues and Software Reuse: An Empirical Study

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

Repeatable experiments, the hallmark of a mature scientific or engineering discipline, are typically absent in software engineering research. Several assumptions are made about the factors affecting software reuse, specifically concerning the role of human factors. This paper describes the results of a controlled experiment designed to evaluate the impact of managerial influence and cognitive abilities on software reuse. The experiment concludes that (1) software reuse promotes higher productivity, (2) reuse resulting from both moderate and strong encouragement promote higher productivity than no reuse, (3) management's strong encouragement to reuse tends to promote improper reuse activities, (4) in general, reuse of a module is unproductive if 30% or less is used for the target system, though as much as 50% can be discarded for some modules and still be worth reusing, (5) of integrative ability, perception speed, and visualization, only the ability to visualize changes made to patterns was related to software reuse, and (6) of the subject's prior experience, only the amount of testing experience was related to software reuse.

# 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 fundamental claims remain unverified by empirical evidence. The potential for reusability to improve the software development process is immense, yet a variety of factors affect its impact. Several theories exist about these factors and the degree to which they 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 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.

These factors are both technical and non-technical in nature. Technical factors include the development paradigm used and the tools available 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.

Human factors, a major influence on the manner in which code is developed [CURB87], play a major a role in software reuse and are, therefore, the primary issue in this research. Without discounting the need for better reuse technology, it is the human mind which develops reusable products and decides if existing code should be reused. Tracz specifically addresses the myth that software reuse is only a technical problem [TRAW88]. A related technical issue, the object-oriented paradigm, was also investigated

in this experiment. The results of the object-oriented aspect of the research are discussed in [LEWJ91].

Two sources of human influence are addressed in the experiment. First, the research explores the role of managerial influence on the reuse process. Second, the experiment investigates the cognitive skills and experience of the reusing programmers. These secondary human factors will not be addressed specifically by the experimental design, but rather as additional analysis as the data permits.

Managerial influence is an external stimulus which affects the software developer in a variety of ways. Management's attitude is consistently mentioned in current literature as an important factor governing the successful reuse process [BIGT87] [TRAW88]. If the manager's attitude is apathetic or weak toward reuse, then it is left to the individual programmers to decide what emphasis is given to reuse. If, however, management's attitude encourages the reuse process, programmers might put more effort into correctly performing reuse tasks. 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.

Differences among individual programmers are known to be a significant factor in programmer production in general. It is natural, then, to question whether specific abilities play an important role in whether a given person can successfully reuse. An individual's cognitive skills could affect how well a programmer evaluates a software component for applicability and integrates it into a new situation. The cognitive skills investigated in this study are:

- Integration: the ability of a person to combine patterns into a unified whole,
- Perception: the speed at which a person detects small differences in patterns, and
- Visualization: the ability of a person to visualize changes in patterns.

These particular skills were chosen because they are intuitively connected to the mental processes employed when reusing software.

Technical experience is another factor investigated. Experts are better at encoding new information than novices [CURB87]. Such differences may also affect the ability to reuse. In the classification scheme presented by Prieto-Diaz, the experience level of the reuser is used as a "fuzzy" modifier of the other factors used to determine which component from a qualifying set should be reused [PRIR87].

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

1) Does any degree of management encouragement to 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?

6) Past what point, in terms of percentage of a component reused for a target system, is it no longer beneficial to reuse?

7) Does a programmer's inherent ability to combine objects into an integrated whole relate to his ability to reuse effectively?

8) Does a programmer's inherent ability to quickly detect small differences in objects relate to his ability to reuse effectively?

9) Does a programmer's inherent ability to visualize changes in objects relate to his ability to reuse effectively?

10) Does a programmer's background experience relate to his ability to reuse effectively?

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. 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 all 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: evaluating potentially reusable products, adapting them to the new situation, and integrating 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 were 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 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 development habits are formed quite early in his professional development. Second, given the number of needed subjects and the amount of control necessary to execute this

experiment, student subjects are the only viable possibility. The efficacy of students as subjects is supported for within-subject experiments by Brooks [BROR80], and therefore applicable to this research.

The research consisted 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. None of the subjects participated in the design or implementation of the reusable components.

## 2.1 Phase One: Component Development

Two sets of potentially reusable components were created during the preparatory phase. To reduce the possibility that the experiment was not biased by 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. Functional equivalence was guaranteed by ensuring that all code meet the same 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 intuitive

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, the subjects, working independently, were assigned the task of implementing the target system. The developed systems 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 such tools would only improve the reuse results and would not negatively affect the other factors being investigated.

The subjects were divided into three groups, pictured as cells in Figure 1. Reflecting different levels of managerial encouragement, each group was given specific guidelines for reuse. One group could not reuse at all, therefore implementing the project using only newly developed code. A second group was encouraged to reuse components as they saw fit. This group was termed "moderate encouragement." The third group was instructed to reuse anything remotely appropriate in the reuse set. This group was termed "strong encouragement." The encouragement was given by written instruction as well as verbally, and was reiterated several times during the course of the project.

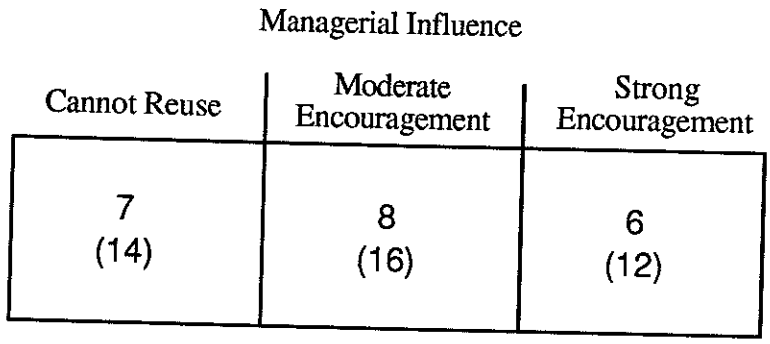


**Figure 1: Number of Subjects (Observations) Per Group.**

Twenty-one subjects were divided into the groups randomly, but were statistically blocked across their computer science grade point averages. 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 target system was divided into two tasks. 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 $\geq 0.2073$). 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 ignored the task factor, effectively creating 42 observations.

## 3. Data Analysis

The data collected during the experiment measures the productivity of a subject during implementation of the target system. Productivity and effort are considered to have an inverse relationship: the less effort expended by a subject to satisfy the requirements of a 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.

Multiple productivity variables are used to obtain a more 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 is needed 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% ($p < 0.05$), an accepted norm. Since our research questions all predict the direction of difference, all tests were performed in a one-sided manner.

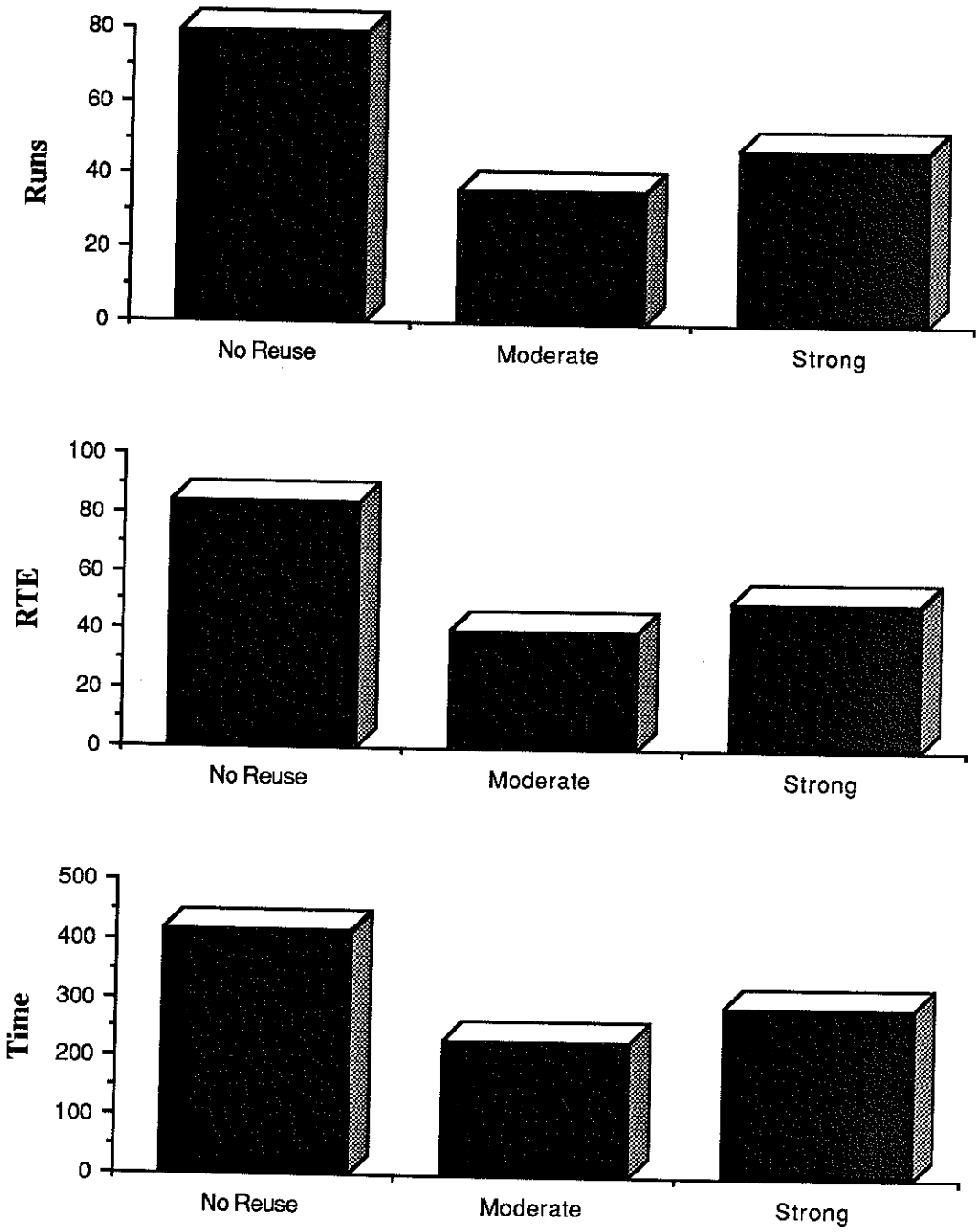


**Figure 2. Group means for primary productivity variables.**

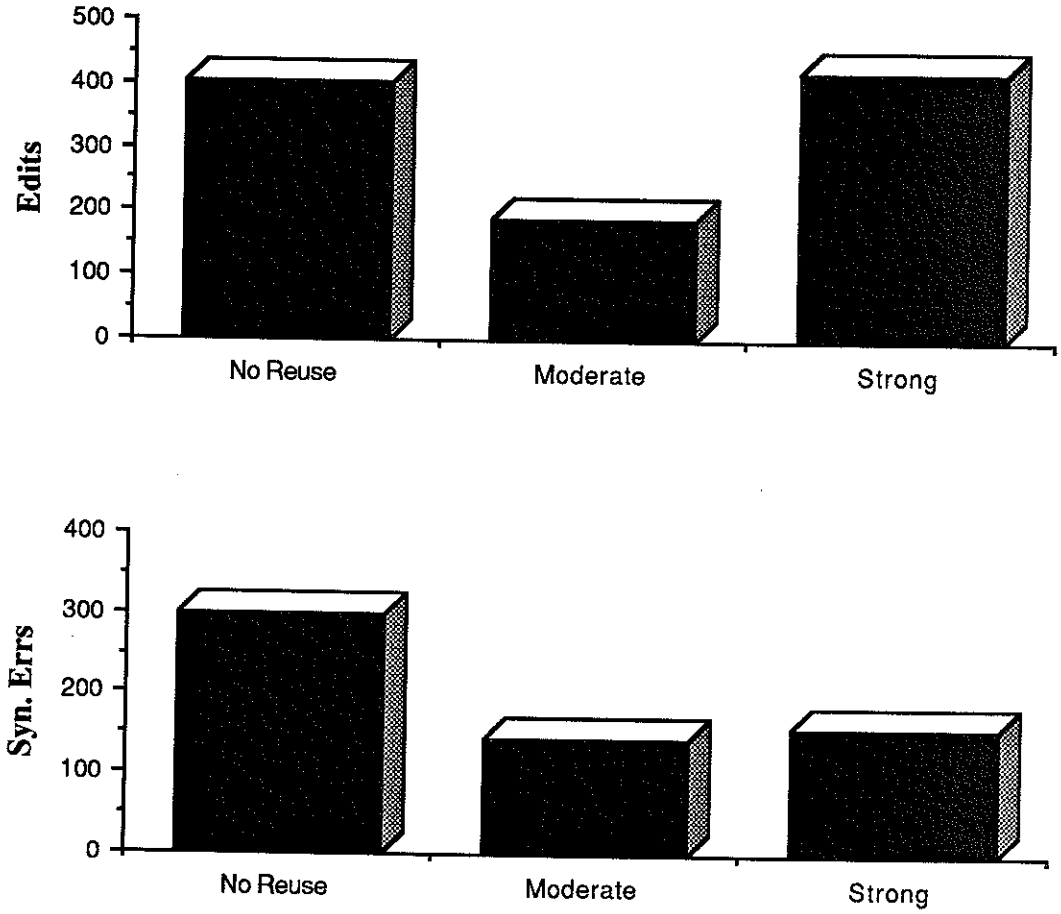


**Figure 3. Group means for secondary productivity variables.**

## 4. Empirical Results

For questions 1 through 4, tables are given which summarize the analyses performed. Questions 5 and 6 deal with issues that do not lend themselves to straightforward statistical analysis given the available data. These questions are analyzed in a less rigorous manner. Finally, questions 7 through 10 address issues that are outside the framework of the group breakdown. These questions are answered using correlation analysis.

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. The questions are divided into three categories: those relating to managerial influence, those relating to cognitive abilities, and those relating to previous experience.

### Managerial Influence

#### 1) Does any degree of management encouragement to 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.

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.

**Table 1. No Reuse vs. Reuse**

| | Significant? | p-value | Means: No Reuse | Means: 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 |

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

The means listed in the fourth column of Table 2 represent the moderate encouragement level of managerial influence. These means are compared to the "no reuse" group, given in column three. Our hypothesis is that the means of the fourth column will be lower than the means of the third 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: No Reuse | Means: Moderate |
|---|---|---|---|---|
| Runs | Yes | 0.0001 | 78.71 | 36.44 |
| RTE | Yes | 0.0001 | 83.79 | 40.75 |
| Time | Yes | 0.0001 | 420.07 | 230.19 |
| Edits | Yes | 0.0001 | 405.71 | 190.81 |
| Syn. | Yes | 0.0001 | 302.14 | 145.00 |

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

**Table 3. Strong Encouragement vs. No Reuse**

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

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 mean of the moderate encouragement group to that 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 | 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 significant, although the means differed in the hypothesized direction. The secondary variables, while differing in the hypothesized direction, were also 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 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 statistically conclude 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 which modules were reused 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 yielded 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.

**6) Past what point, in terms of the percentage of a component reused for a target system, is it no longer beneficial to reuse?**

Given the nature of the data, it is impossible to answer this question with a single percentage value. However, examination of the modules reused other than those in the base set (discussed in the previous question) yields an indication of the level past which it is no longer beneficial to reuse.

The percentage in question can be approximated by examining the set of modules used by the strong encouragement group minus the modules in the base set. From question 5, it is this set of modules which caused the lower productivity in the strong group. Excluding the modules in the base set, the maximum percentage reused by subjects in the strong group was 70%, and the minimum reused was 30%. Therefore, the point past which it is not worth reusing a module is between these limits.

However, this range can be refined. Many subjects reused the base set modules at levels ranging from 50 to 100 percent reuse. Given the higher productivity of the moderate encouragement group, which concentrated on the base set, these reuse practices seem beneficial. The minimum percentage reused of any module in the base set by either reusing group was 50%, indicating that some modules are worth reusing if even only half can be used for the target system.

Therefore, the delimiters for the beneficial cut-off point are between 50 and 30 percent. That is, it is certainly not productive to reuse a module if only 30% can be salvaged. However, some modules should be reused if as little as 50% can be used.

## Cognitive Abilities

The following questions (7 through 9) are addressed by the analysis in Table 5, which shows correlations between the productivity variables and the scores of the cognitive tests. Keep in mind that the productivity variables are actually measures of effort, therefore we expect negative correlations between these variables and the scores of the cognitive tests. The column headers are the test designators as taken from the battery of cognitive tests in [EKSR76]. The nature of these tests is elaborated upon in the discussion below.

**Table 5. Cognitive Test Correlations**

| | IP-1 | IP-2 | P-2 | P-3 | VZ-1 | VZ-2 |
|---|---|---|---|---|---|---|
| Runs | 0.12 | - 0.05 | 0.21 | 0.23 | - 0.22 | - 0.32 |
| RTE | 0.12 | - 0.26 | 0.22 | - 0.15 | - 0.51 | - 0.38 |
| Time | 0.26 | - 0.16 | 0.09 | - 0.13 | - 0.61 | - 0.22 |
| Edits | 0.30 | 0.19 | 0.27 | 0.46 | 0.01 | - 0.12 |
| Syn. | 0.21 | 0.22 | 0.22 | 0.49 | 0.05 | - 0.24 |

**7) Does a programmer's inherent ability to combine objects into an integrated whole relate to his ability to reuse effectively?**

The first and second columns of Table 5 represent the two tests measuring the integrative processes of the subjects (tests IP-1 and IP-2). Since the scores are correlated against the productivity measures, our hypothesis is that the correlation values will be negative.

All correlation values for the integrative processes tests are within -0.26 to 0.30. Not only are the directions of the correlations inconsistent, but the values are generally close to zero. No relationship between the cognitive ability to integrate objects into a unified whole and the ability to reuse effectively is demonstrated by these data.

**8) Does a programmer's inherent ability to quickly detect small differences in objects relate to his ability to reuse effectively?**

This question deals with the subject's perception speed. The third and fourth columns of Table 5 show the correlation values for the two tests measuring this ability (tests P-2 and P-3). Our hypothesis is that if any relationship exists, the correlations will

be negative, indicating that as perception speed increases, the ability to reuse effectively increases (and thus effort decreases).

Again, as with the previous question, the correlation values are inconsistent and very close to zero. No relationship between the cognitive ability to quickly detect differences in objects and the ability to reuse effectively is demonstrated by these data.

**9) Does a programmer's inherent ability to visualize changes in objects relate to his ability to reuse effectively?**

The fifth and sixth columns of Table 5 list the correlation values for the tests measuring visualization, or the ability to mentally manipulate objects (tests VZ-1 and VZ-2). Our hypothesis, once again, is that the correlations will be negative.

This relationship is given some support by the data. The fifth column shows two of the main productivity variables, RTE and Time, with fairly large negative correlation values. While the sixth column's values are not as demonstrative, all primary variables for the two tests are consistently negative. Given the nature of the data represented, these results indicate that some relationship exists between the ability to visualize changes in objects and the ability to reuse effectively.

## Previous Experience

**10) Does a programmer's background experience relate to his ability to reuse effectively?**

Several variables representing background experience in activities which could relate to the reuse process were measured for each subject. These variables were collected separately for academic and professional experience. Several variables, including all variables measured for professional experience, did not show differences between subjects due to lack of experience. Therefore, no correlations were possible for these measures. Table 6 lists the correlation values for the variables which had sufficient merit to analyze.

The selected data recorded the number of months of experience each subject had (1) writing in Pascal, (2) writing in C, (3) creating external documentation for software systems, (4) performing testing activities, and (5) using the Unix operating system. Our hypothesis is that, if a relationship exists between experience and the ability to reuse effectively, the correlation values will be negative. This implies that the more experience a programmer has, the more effective he will be at the activities involved in the reuse process.

**Table 6. Background Experience Correlations**

| | Pascal | C | Doc. | Testing | Unix |
|---|---|---|---|---|---|
| Runs | - 0.03 | - 0.15 | 0.03 | - 0.31 | - 0.26 |
| RTE | - 0.43 | - 0.23 | 0.29 | - 0.54 | - 0.11 |
| Time | - 0.40 | - 0.17 | 0.37 | - 0.59 | - 0.01 |
| Edits | 0.44 | - 0.02 | - 0.06 | 0.10 | - 0.24 |
| Syn. | 0.47 | - 0.20 | 0.02 | 0.16 | - 0.14 |

The first and second column show the relationship between experience using a given language and effective reuse. While the correlations with primary variables in both columns are consistently negative, the values are not high enough to indicate a meaningful relationship. The same conclusion is drawn for the Unix experience variable, as shown in the last column. The documentation variable, shown in the third column, did not even demonstrate a correlation in the hypothesized direction and, like the others, had values too close to zero to indicate a relationship.

Only the amount of experience testing, shown in the fourth column, showed any relationship to effective reuse. All three primary productivity measures were negatively correlated with the testing variable, and two of them (RTE and Time) had correlations less than -0.5. These results indicate that a relationship exists between the amount of experience a subject has in testing software products and the ability to reuse effectively.

## 5. Summary

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

(1) Software reuse promotes higher productivity (question 1),

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

(3) Management's strong encouragement to reuse does tend to promote improper reuse activities (questions 4 and 5), and

(4) In general, reuse of a module is unproductive if 30% or less is used for the target system, though as much as 50% can be discarded for some modules and still be worth reusing (question 6),

(5) Of integrative ability, perception speed, and visualization, only the ability to visualize changes made to patterns was related to software reuse (questions 7, 8, and 9), and

(6) Of the previous experience reported by the subjects, only the amount experience testing code was related to software reuse (question 10).

The practical application of this study for management is clear. Reuse increases productivity and therefore should be encouraged. However, there is evidence that too much encouragement can lead to improper reuse, therefore a moderate level of encouragement is suggested. The process of selecting candidate components for reuse can be guided by the general rule that 50% reuse is usually warranted, and less than 30% reuse is unwise. Finally, tests of visualization and amount of testing experience are good indications of an individual's ability to reuse.

## 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.

[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).

[PRIR87] 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.