PROGRAMMER PERFORMANCE AND THE EFFECTS OF THE WORKPLACE

Tom DeMarco and Tim Lister

The Atlantic Systems Guild
353 W. 12th Street
New York, NY 10014 USA

Abstract

Wide variation in programmer performance has been frequently reported in the literature [1, 2, 3]. In the absence of other explanation, most managers have come to accept that the variation is due to individual characteristics. The presumption that there are order-of-magnitude differences in individual performance makes accurate cost projection seem nearly impossible.

In an extensive study, 166 programmers from 35 different organizations. participated in a one-day implementation benchmarking exercise. While there were wide variations across the sample, we found evidence that characteristics of the workplace and of the organization seemed to explain a significant part of the difference.

Keywords: Management Issues, Productivity, Programmer Workplace.

1. Introduction

It is common wisdom that there is a huge variation in individual performance rates among programmers. Sackman, Erikson and Grant [1], for example, reported differences of as much as 25:1 in the time required for a given programming task. Boehm [2] reported that the cost driver derived from individual characteristics of team members was nearly twice as large as the second largest driver. And Augustine [3] observed that more than 50% of the work is typically done by 20% of the people.

While most programmers and managers accept that there are such variations, few have any idea where they themselves stand on the performance scale. Evidence that a particular person performed in the upper ten percentile across the industry would lead to elation; placement in the lower 10 percentile would lead to despondency — *but neither would cause great surprise.* Software developers (and the organizations they work for) remain largely ignorant of their own capacities.

The issue of performance variation is further complicated by the suspicion that high and low performers tend to cluster in different organizations. In [4], Putnam indicated success in ascribing a *technology factor* to organizations in order to explain their varying capacity. The technology factor was empirically derived, based on past performance. Once set for an organization, it was surprisingly constant across different projects and different teams. The usefulness of Putnam's technology factor implies that the wide differences in performance may be more a function of corporate culture and the workplace than of inherent individuality.

If there are large differences between organizations, managers cannot afford to remain ignorant of where they stand. If the differences can be even partially explained by remediable characteristics of the environment and the workplace, then those characteristics deserve to be a major focus of our productivity improvement effort.

2. The Coding War Games: A Public Benchmarking Exercise

The 1984 Coding War Games were conceived as an opportunity for individuals to find out how they performed relative to a sample of their peers. A further objective was to test hypotheses about the effect of the workplace.

The exercise was run according to the scheme of [5] as an open competition. Each organization submitted one or more competition pairs, each pair made up of two volunteer implementors. The two members of each pair performed the same one-day implementation task, working to the same specification. They were encouraged to compete with each other as well as with the rest of the sample. They worked at their own workspace (office, cubicle, terminal room, etc.), during normal working hours. They used the normal languages, machines and development facilities provided by their organizations. The two members of each pair worked in the same language and used the same support environment.

A total of 166 programmers from 35 organizations participated in the 1984 games. Characteristics of the participants are described in Figures 1 and 2.

The exercise was implementation of a program to a rigid specification. The program involved syntactic and semantic edits on an input stream of calendar dates, followed by computation of day-intervals between specified dates as much as 8 centuries apart. The average program built in compliance with this specification was 220 lines long. Two thirds of the programs were between 133 and 297 lines in length. The average COBOL program was 237 lines. The specification called for on-line operation, but there was an alternative for batch operation which was used by approximately 5% of the pairs.

The project was organized in such a way that there were two well-defined milestones: 1) Clean compile/ready for test, and 2) All work complete. The entire sample was divided at the beginning into two "rounds." Both rounds used the same exercise, but they used slightly variant test procedures. Round One programmers did no testing on their own code; each coder completed and desk-checked his/her program and, after producing an acceptable clean compile, then gave it to the other pair member to test. Round Two programmers ran the exercise with the more familiar procedure of coding and then testing their own code. Because the different testing approaches gave a slightly different meaning to the Milestone 1, data from the two rounds was compiled separately. Round One was composed of the first 100 entrants.

After programmers had completed their own testing, they ran a pre-set acceptance test provided in sealed form with the instruction kit. The acceptance test consisted of ten sample inputs. Participants recorded the outputs on a form which was returned to the organizers for analysis. The acceptance tests were designed to measure coherency, accuracy, precision, robustness and edit efficiency of the programs.

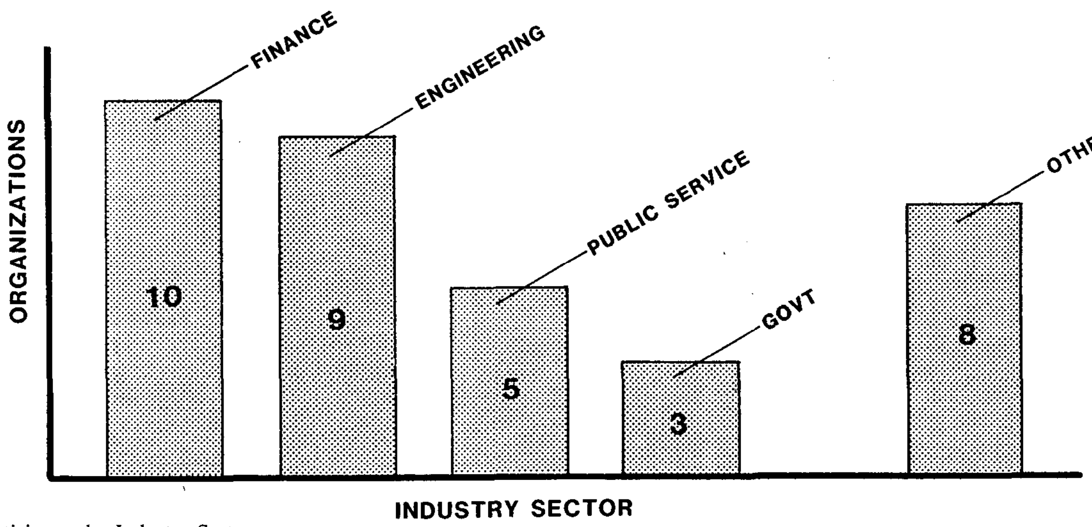

Figure 1. Participants by Industry Sector

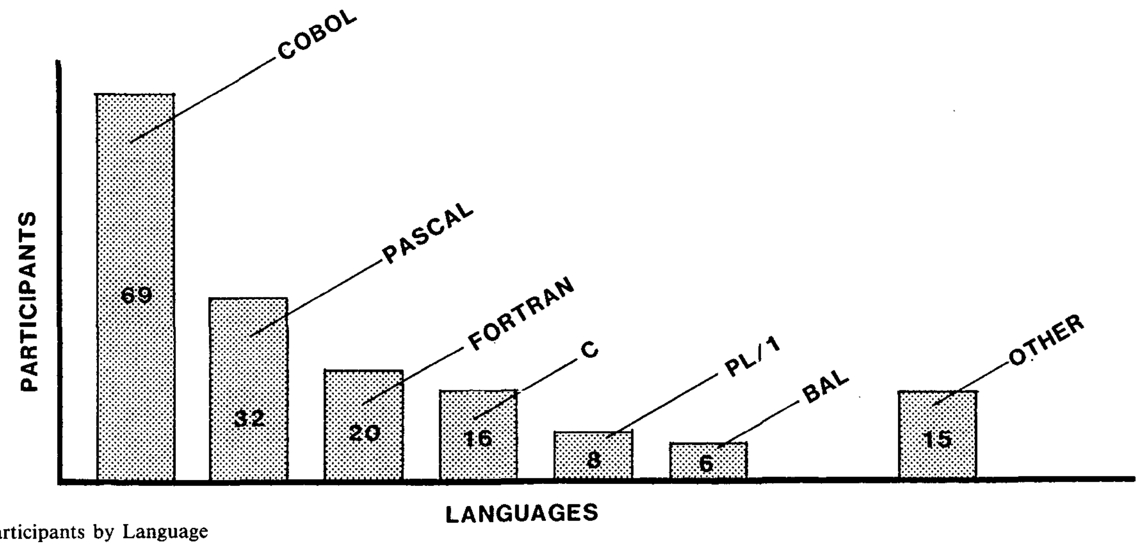

Figure 2. Participants by Language

During the exercise, participants kept rigorous track of time spent. They noted periods of work, type of work, periods of interruption and nature of each interruption. All this time data was recorded on a form supplied and returned to the organizers.

Each participant filled out an extensive questionnaire designed to determine measurable characteristics of the workplace and environment as well as the individual's subjective attitude toward these factors.

3. Programmer Performance Across the Sample

Figure 3 shows how participants performed (how much work time they used) in reaching Milestone 1. There is indeed a considerable difference across the sample: The variation from best to worst is a factor of 5.6 to 1. Average performance was 2.1 times slower than the best. The half above the median outperformed the half below the median by 1.9 to one.

While variation over the entire group was pronounced, there was very little variation within each pair. Consider these results:

- The overall fastest performer was paired with the second fastest.
- The overall slowest performer was paired with the second slowest.
- Of the thirteen participants that did not finish the exercise, all but three were paired with other non-finishers.

The two members of each pair were arbitrarily coded Red and Blue at the beginning of the exercise. The performance of one pair member turned out to be a strong predictor of the performance of the other. This is shown in Figure 4, where Red performance is plotted against Blue. Each point on the graph represents one pair's joint performance. (The x-coordinate gives Blue and the y-coordinate gives Red.) A point on the 45 degree line would indicate a pair in which the two participants performed identically. The correlation coefficient between Red and Blue performance was 0.79.

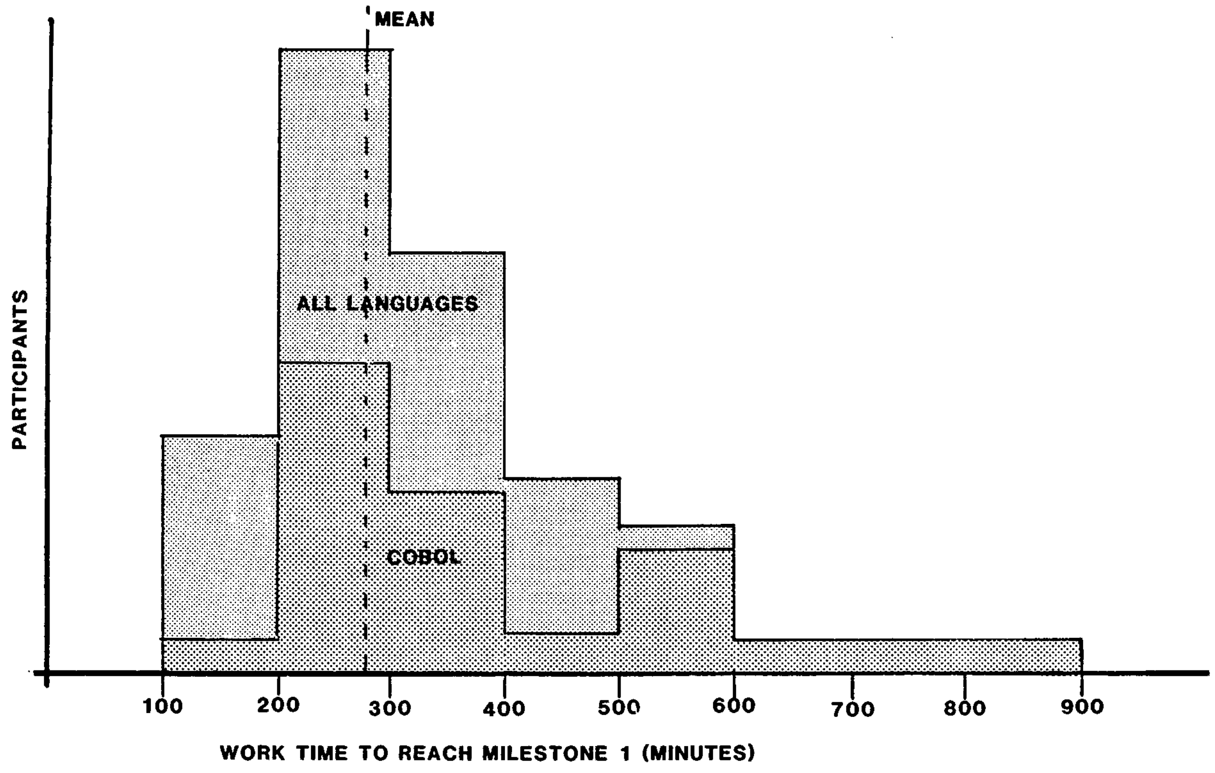

Figure 3. Spread of Performance

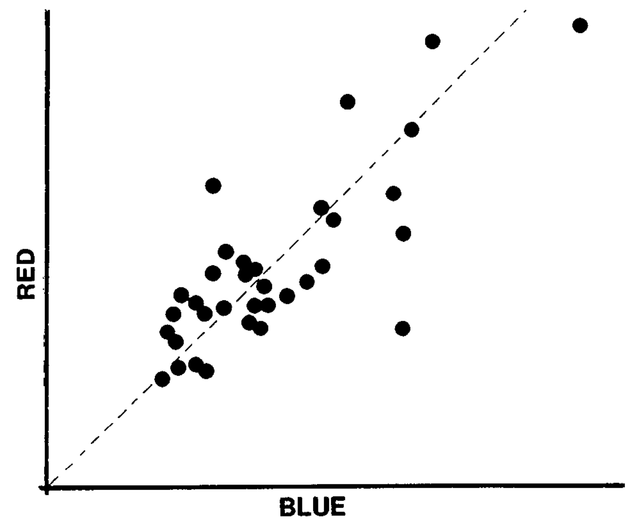

Figure 4. Correlation Between Teammates (Round One Data)

The marked grouping of points around the 45 degree line suggests that characteristics of the workplace and corporate culture (which are constant for the two members of each pair) may explain much of the overall variation in programmer performance. The difference within a pair is an indication of truly individual performance capacity. For the average pair in the sample, the difference between the two performers was a factor of 1.21 to one. For 80% of the pairs, performance of the two members was within 34% of each other.

Figure 5 shows the quality of the resultant programs as judged by a sample of five key acceptance tests. Though they were not allowed to debug their programs at all, more than one third of Round One participants produced a program that passed the major tests of coherency, functionality and accuracy on its first run.

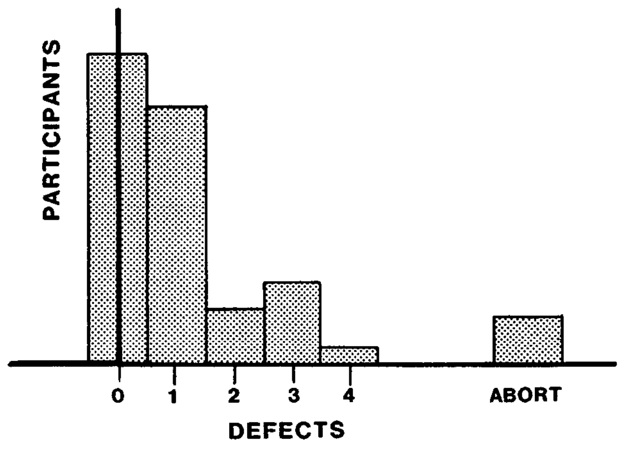

Figure 5. Spread of Defect Counts

Figure 6 shows the relationship (actually, the lack of relationship) between time to reach the milestone and number of defects. There appeared to be no quality penalty for rapid performance: Faster than median performers had fewer defects than slower than median performers, and those who performed in the top 25% in terms of time to reach the milestone had an average defect density 30% lower than that of the rest of the sample.

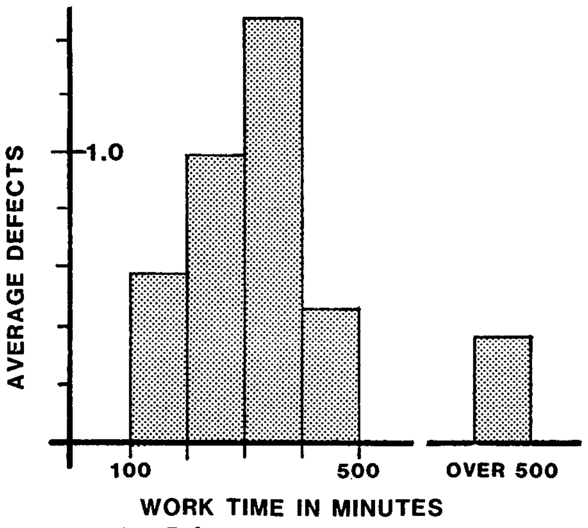

Figure 6. Speed vs. Defects

4. Characteristics of the Programmer Workplace

It may be impossible to quantify the illusive concept of "corporate culture," and difficult to improve it substantially. But this need not be true of the programmer's workplace and its surrounding environment. In this section, we shall set out observed characteristics of the workplace and in the next section we shall show how such characteristics were correlated to performance.

The environmental questionnaire that each participant filled out asked for both objective data about the workplace (how much space, what provisions for privacy etc.) and subjective assessment by the participant ("Does your office space make you feel appreciated?"). From data taken from all 160 respondents, the following picture emerges of the prototypical programmer workplace:

PROGRAMMER WORKPLACE
(AVERAGES FROM THE SAMPLE)

Dedicated space:	63 square feet
Enclosure:	cubicle walls (78%)
Cubicle height:	5 feet
Dedicated terminal:	60%

There were substantial variations in these characteristics across the sample. Figure 7, for example, shows the variation in dedicated floor space.

A rather grim picture of the workplace emerges from the subjective assessments by participants:

QUESTION	PERCENT OF RESPONDENTS
Is your workplace acceptably quiet?	58% No
Is there sufficient privacy?	61% No
Do people often interrupt you needlessly?	62% Yes
Is it difficult or impossible to work effectively in your workplace from 9-5?	41% Yes
Does your workplace make you feel appreciated?	51% No
Is your workplace at work as pleasant as your workplace at home?	54% No

Reading through the respondents' free-form comments about the environment is a distressing experience. Many programmers appear to be continually frustrated in attempts to work. They are plagued by noise and interruption, and pessimistic that the situation will ever be improved. The data recorded about actual interruptions supports the view that the so-called "work-day" is made up largely of frustration time. Reproduced below is a portion of one typical time-sheet from the exercise:

WORK PERIOD FROM – TO	TYPE OF WORK	WHAT INTERRUPTION CAUSED THE END OF THIS WORK PERIOD?
2:13 - 2:17	Coding	Phone call
2:20 - 2:23	Coding	Boss stopped in to chat
2:26 - 2:29	Coding	Question from colleague
2:31 - 2:39	Coding	Phone call
2:41 - 2:44	Coding	Phone call

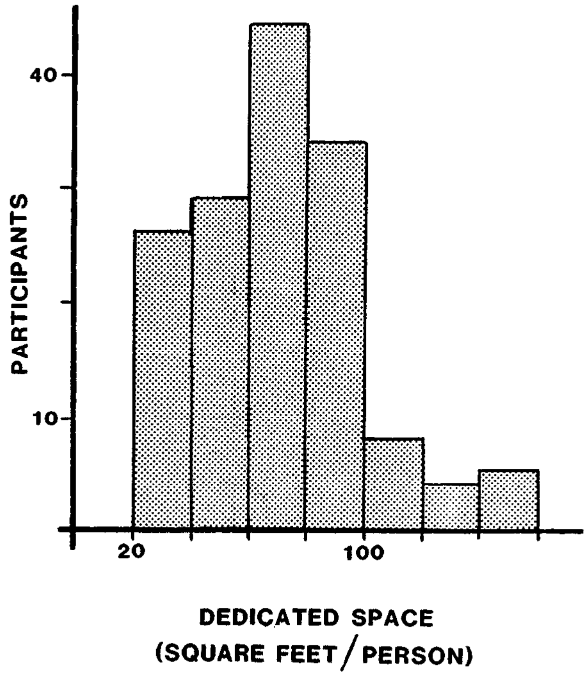

Figure 7. Variation of Dedicated Floor Space

5. Effects of the Environment on Performance

In order to detect any correlation between environment and performance, we divided the set of finishers into four groups based on performance. Average performance of those in the upper 25% was 2.6 times better than that of those in the lower 25%. We then compared environmental factors recorded by the top 25% of performers to those of the lower 25%. A sample of results is presented below:

ENVIRONMENTAL FACTOR	TOP 25%	BOTTOM 25%	ALL
Dedicated floor space	78 sqft.	46 sqft.	63 sqft.
Acceptably quiet workspace	57% yes	29% yes	42% yes
Acceptably private workspace	62% yes	19% yes	39% yes
Can you silence your phone?	52% yes	10% yes	29% yes
Can you divert your calls?	76% yes	19% yes	57% yes
Do people often interrupt you needlessly?	38% yes	76% yes	62% yes
Does your workspace make you feel appreciated?	57% yes	29% yes	45% yes

We concluded that the two groups work in significantly different environments. The top performers are in fairly generous space that manages to protect them from at least some distractions. The telephone interruption problem has been addressed to the point that most people know their phones will be picked up by a clerical worker if ignored. The space is relatively pleasant and a culture of interrupt consciousness has evolved. People have relatively long periods of interrupt-free work.

The bottom 25% work in tiny cubicles — eight of this group reported dedicated space of 40 square feet or less! The phones ring until answered and cannot be diverted. There is little or no interrupt consciousness, with managers among the worst offenders. (One participant wrote, "My boss switches his secretary's phone to me when she's out.") People are forced to work in short periods of time between interrupts.

There is a danger here of confusing cause and effect. We have implied, for instance, that better (quieter) workspace may result in higher productivity. But it may be that high productivity has been rewarded by more floor space. Hence the best performers may have gravitated naturally to the more commodious and thus quieter space. To investigate this possibility, we analyzed three organizations that had submitted nine or more teams each. Within each of these organizations we found little or no variation in the main environmental factors. The best performers from each of the companies worked in more or less the same floor space and same noise level as the worst performers.

As a further control, one organization had 18 programmers take part in the exercise, working in their normal workplace while another six worked in a specially contrived "clean room" environment, free from most interruptions and noise. The clean room group out-performed their peers by 40% in the time required to reach the milestone.

Whether a better workplace causes higher productivity or higher productivity workers gravitate toward organizations with a better workplace should be of little concern to the software manager. Either argues for more concentration on the physical environment in which programmers (try to) work.

6. Summary: The Case for Improving the Programmer Workplace

In spite of widespread productivity consciousness, our industry has tended to ignore the effect of the workplace. The typical productivity manager considers changes to the physical characteristics of the workplace to be outside his/her charter. Our findings imply that the productivity charter should be widened: Environmental factors such as noise, privacy and interruptibility may be keys to substantial productivity improvement. For example, changing the workplace at a given company from that of typical of a "lower-25% performer" to that of an "upper-25% performer" offers the potential of a 2.6 to one improvement in the time to perform a complex programming activity.

REFERENCES

[1] Sackman, H., W.J. Erikson and E.E Grant.Exploratory Experimental Studies Comparing Online and Offline Programming Performance." *Communications of the ACM,* Vol. 11, No.1 (January, 1968), pp 3-11.

[2] Boehm, B.W. *Software Engineering Economics.* Englewood Cliffs, N.J.: Prentice-Hall, 1981.

[3] Augustine, N.R. "Augustine's Laws and Major System Development Programs." *Defense Systems Management Review, 1979,* pp 50-76.

[4] Putnam, L.H. "A General Empirical Solution to the Macro Software Sizing and Estimating Problem." *IEEE Transactions on Software Engineering,* Vol. SE-4, No. 4, July 1978, pp 345-361.

[5] DeMarco, T. *Controlling Software Projects: Management, Measurement and Estimation.* New York: Yourdon Press, 1982.