Does money motivate software engineers to improve the quality of their work?

Chris Brown
Department of Computer Science
North Carolina State University
Raleigh, NC
dcbrow10@ncsu.edu

Effat Farhana
Department of Computer Science
North Carolina State University
Raleigh, NC
efarhan@ncsu.edu

ABSTRACT

Motivation, although difficult to quantify, is considered a key factor in software development quality. Motivating software engineers is becoming increasingly important as our society becomes more dependent on technology and the problems we need to solve become more complex. This paper describes an observational study to discover how financial incentives influence the performance of software engineers. The results also give some intriguing insights into whether money is equally the same motivating factor for different tasks that vary in cognitive levels and require different problem-solving strategies to solve.

Keywords

Motivation; Software Development; Software Engineers

1. INTRODUCTION

One of the most significant problems that companies are facing today is finding good workers and then keeping them motivated after they are hired. Bowditch and Buono noted that one of the most important problems for organizations concerns *why* workers perform and behave the way they do at their jobs [4]. Companies want to know what makes people work hard and how they can influence the people to have positive attidues about their jobs, which is directly correlated to their performance and the quality of the product or service produced from their work. Vroom writes in *Work and Motivation* that there is a strong relationship between humans and their work, noting that it is necessary for employees to prefer working over not working, and created a model to present a given person's performance as a function of how motivated they are and their individual ability (Performance = f(Ability x Motivation) [17].

Career analyst and best-selling author Dan Pink argues that science has shown that the current methods of motivating employees, such as offering raises or bonuses, do not work and a new approach is needed [11]. He notes that financial incentives are not only outdated and ineffective, but can actually be harmful to the workplace environment and the overall economy as 21st century businesses have increasingly complex and difficult problems to solve that require more creative solutions.

The software engineering industry specifically has become a vital part of our society with a variety of programmed technology playing a crucial role in our everyday lives. The National Institute of Standards and Technology notes that software has become a crucial part of developing, producing, marketing, and supporting products and services for nearly every business sector in the U.S. [15]. Gartner Research estimates that a total of $3.54 trillion dollars will be spent on the IT industry worldwide in 2016 and that number will continue increasing as the demand for software rises [19].

Unfortunately, software engineers are also one of the hardest groups of workers to keep motivated. The Software Engineering Institute reported that for a given software engineering project, an average of 70% of the total costs goes towards human resources and keeping the developers happy. In 2008, the turnover rate for software engineers was above 20% and a typical programmer would stay at a company for an average of only 23 months [16]. Our research will look to see if monetary incentives can motivate computer programmers and improve their performance in completing software engineering tasks as our society becomes increasingly reliant on technology.

2. RELATED WORK

Motivation has been identified as a key factor affecting many important aspects of software development such as productivity, adherence to budgets, increases in staff retention, and reduced absenteeism [13]. Previous studies have found that people working in the software industry are motivated by the nature of the job [5], technical challenge [14], and problem solving. In addition, a predominant perspective in motivation research is that of the organization focusing on issues such as turnover [1], performance [6], working in a renowned company, and creating a meaningful product [8]. These factors were found to be highlighted by more programmers than financial incentives. Whitaker noted the difficulty of motivating and keeping software developers and concluded that clearly defined roles and rewards can help keep developers happy, emphasizing companies should use rewards after a goal has been met rather than giving incentives such as bonuses or raises that are expected [18].

Many studies have also been performed across various dis-

DOI: 10.1145/1235

ciplines to uncover the relationship between motivation and performance. Economist Dan Ariely performed a set of experiments where subjects were split into three groups that would either receive small, moderate, or large rewards for completing different tasks [2]. His study concluded that "higher incentives led to worse performance". In contrast, Human Resources Management did a study demonstrating the gap between what people say and what they do with respect to pay [12]. When asked directly about the importance of pay, people tend to list it as the fifth most important motivating factor in lists of potential motivators, but when behavioral responses to changes in pay were taken into account salary was found to be the most crucial incentive.

Our project is based on a 1962 psychology study performed by Sam Glucksberg, who wanted to determine the influence of incentives on problem solving [9]. Glucksberg divided participants into two groups for two different experiments, a "high-drive" group that was told ahead of time that they would receive a monetary reward and a "low-drive" group that would not be compensated. He found that the high-drive group performed well in a basic version of task that did not require much critical thinking, but they performed much worse than the low-drive group in a problem solving task that required a creative solution. Our experiment implements a theoretical replication of Glucksberg's study to discover how monetary incentives specifically impact the performance of software engineers and their ability to complete software development tasks requiring creative and non-creative solutions.

3. METHODOLOGY

We aimed to investigate how monetary incentives affected the performance of software developers in performing tasks. Our study is a theoretical replication of a psychology study by Glucksberg geared specifically towards software engineers and programming tasks [9]. We wanted to determine:

RQ1: Do financial incentives impact the performance of software engineers? and

RQ2: Do monetary rewards have a different impact on software developers when solving more creative problems?

We had participants perform two debugging tasks and measured if financial incentives had an impact on their performance in completing the tasks.

3.1 The Two Tasks

Glucksberg performed two different experiments to examine how incentives impacted two different types of problem solving strategies, functional fixedness and perceptual recognition. Functional fixedness was a term introduced by Karl Duncker used to describe a cognitive bias that prevents people from using an object other than the way it is traditionally used [7]. People must overcome functional fixedness in order to find a solution to more complex and creative problems. On the other hand, perceptual recognition involves using perception and the senses to solve a problem.

For our study, we decided to use pseudocode debugging tasks for the evaluation rather than have participants solve programming problems by writing code. Pseudocode was used to avoid any bias and prevent the participants' previous experiences and knowledge of a particular programming language from impacting the results. We decided to use debugging because it has become the most important activity in software development, with finding, comprehending, and fixing bugs taking up 70-80% of software engineers' time [10]. Additionally, debugging is a programming activity that can require both of the types of problem-solving examined by Glucksberg in his study. Perceptual recognition is required to search through the source code and find a mistake. Functional fixedness can also be used to find bugs in the code in unexpected places and can help developers come up with an efficient solution to fix the error without breaking other parts of the program.

One of our tasks was categorized as non-creative because it was a simple debugging task with a very straight-forward solution (Appendix B). The error is easy to find perceptually, but it was also very easy to overlook if subjects did not look carefully enough. Participants could go through the code linearly and, based on the specifications we gave them, would find that there is a typo in line 5 when the program checks the length of the password. The program should return false when the password length is less than 8, but instead there is a greater than sign.

The other task was more creative debugging task, which required a deeper understanding of the program logic to find and fix the bug (Appendix A). The error is not as easy to discover, and appears in line 20 of the code. Functional fixedness comes into play here because `count++` is an operation normally used for incrementing the value of a number in computer programming. However, the `++` operator is a *postfix increment operator* which is used correctly in this program. Bloch and Gafter point out in Java Puzzlers that running a similar program in Java will always return zero because `count = count++` is equivalent to doing

```
int tmp = count;
count = count + 1;
count = tmp;
```

or saving the initial value of count, incrementing the value of count, and then resetting count back to the original value [3]. There is also an extra factor in that undergraduate students taking classes towards an accelerated master's degree are counted as graduate students. Participants would not be able to easily figure out the error using perception alone, and would have to work through the code and possibly try an example to see the error. There are also multiple possible solutions to fix the bug in the code, but the simplest one is to replace line 20 with just `count++`. We designed these two debugging tasks to find out if financial incentives drive the quality of problem solving and if it also had an impact on the type of strategies required to solve the problem.

3.2 Participants

Six participants showed interest in taking part in our study. We conducted a pre-survey for each participant before performing our study in order to get demographic and background information for each person. All of the subjects graduate students at NC State University with previous computer programming and debugging experience, and have worked in various roles such as software developer, tester, architect, and manager for different companies such as Amdocs, Bank of America, Wells Fargo, Samsung, Synopsis, WeightWatchers, NC State, creating a start-up, and more.

Table 1: Participant Demographics

Participant	Academic Exp. (years)	Industry Exp. (years)	Group
P1	6	6	1
P4	8	2	2
P5	5	2	1
P6	6	2	2
P7	6	0	1
P8	5	14	2

* P2 and P3 did not come at scheduled time for study.

Table 1 provides additional information about of the participants who took part in our study.

All of our participants had a technical undergraduate bachelor's degree in computer science or electrical engineering. Participants had mean professional experience of 4.3333 years (SD=5.1251) ranging from 0 to 14. After filling out the pre-survey, participants were placed into one of two different groups to complete the study. Group 1 performed the non-creative task first and then were offered a monetary reward to complete the creative task, and Group 2 completed the creative task first and was then offered money for the non-creative task. We did this to determine if financial incentives not only had an impact on the performance of software developers, but also if it influenced how they use different problem-solving strategies to solve software engineering problems.

3.3 Procedure

To complete the tasks, participants were provided pens and paper with a pseudocode program to figure out their solution and use sketches if needed. We wanted our study to represent a real-world situation, so the researchers acted as managers of a software company developing an educational application to store information about the students in a school. We handed over the tasks to each participants and then instructed each participant to debug two pseudocode programs and come up with a solution to fix it. Participants were given a maximum of 10 minutes to complete the task and were unaware that rewards were provided as part of out study. For each person we asked them to complete the first task without offering them any incentives. Before completing the second task, participants were told that the top performer would get $25 and the participants who finished in the top 25% would get $5 based on the efficiency and accuracy of completing the task.

3.4 Analysis Methodology

After completion of the two tasks, each participant completed a semi-structured post-interview session to get more information on specifically how the monetary incentives influenced how they solved the first problem compared to the second one and discover what their motivations were as a software developer. Task completion time and a correct solution were used to determine the best participant and the top 25% of performers. Statistical analysis was performed based on two performance measures, efficiency and accuracy, to help determine how financial incentives impacted the performance in completing our tasks.

4. RESULTS

This section will present the results of our evaluation divided into quantitative results analyzing the difference in times of the subjects between the two tasks and qualitative results based on the post interviews with each participant.

4.1 Quantitative Results

The time to complete the tasks were measured in minutes. For each task category, we define the two "Drive" levels as follows:
Drive=High: Offered money for performance.
Drive=Low: No monetary incentive for performance.

The mean, standard deviation of solution time for each drive levels and the total number of failures are presented in Table 2.

Table 2: Solution Time to Solve Tasks

Task Category	Drive	No of Failure	Solution Time (Mins.)	
			Mean	SD
Creative	High	1	6.267	2.646
	Low	0	5.842	0.335
Non Creative	High	0	4.606	2.852
	Low	0	0.999	0.311

We also performed unpaired t-test on solution time for both task categories. The null hypothesis was:

H_0: Monetary incentive is unrelated to performance of software engineers.

Table 3: Unpaired t-test Result

Task Category	t-value	P value	Accept/ Reject H_0
Creative	0.276	0.796	Accept H_0
Non Creative	2.177	0.095	Accept H_0

From Table 2, it is evident that offering money actually had degraded the performance of participants of both task categories. The result of t-test also aligns with post interview result responses. We have to accept the null hypothesis H_0, that offering monetary benefit has no impact on the quality of performance.

4.2 Qualitative Results

After completing the experiment, we conducted a short semi-structured interview with each participant in order to get more information about their computer science background and ask about the study itself. All of the participants mentioned they had previous experience with debugging code and solving creative problems using computer programming. Additionally, all of the participants that had professional development experience received a raise or bonus while they were employed. None of the participants thought that our debugging tasks were unfair or too difficult to solve.

Most participants mentioned that offering a financial incentive had some impact on their ability to solve the debugging task we provided. The most popular response we

received was that the monetary reward made the participant work faster than they wanted and not at their own pace (P1, P4). P6 said that it made her "more focused". P8 also mentioned that the reward had an impact on how he solved the second task and he was faster. Money put more pressure on participants to perform and solve the problem quicker but impacted their performance.

Our interviews also showed that money is not one of the main motivators for software developers. We asked participants what motivated them to study computer science and work as a software engineer, and only one subject said his main motivation for working in industry was "to earn a living" (P8). Most of the answers we received were interest in technology and programming, a desire to work on interesting projects, creating cool products, working with "cutting edge technology" (P5), an ability to learn and apply specific computer science concepts such as machine learning (P1) and artificial intelligence (P5), etc. This shows us that the most important motivating factor for software engineers is the work itself rather than extrinsic motivators such as monetary incentives.

5. DISCUSSION

Our study gave some insights into our research questions, but we were unable to reject our null hypothesis or get significant results.

5.1 Do financial incentives impact the performance of software engineers?

Our first research question wanted to discover if offering money had an impact on the performance of developers performing software engineering tasks. RQ1 would have implications in industry because most companies use financial incentives such as raises or bonuses to encourage their employees to perform at a higher rate and keep them happy, but that may not be the best method for motivation. Our results showed that, on average, monetary rewards had a negative impact on the performance of our participants.

The total average time to complete the high-drive tasks was 5.437 minutes while low-drive tasks took 3.421 minutes, however this difference in the times was not statistically significant. The most important contribution to this research question comes in the qualitative results, where four participants mentioned that offering the monetary reward had some type of impact on how they solved the second task. Additionally, only one person cited money as a motivation to work in industry while everyone else was more motivated by the projects themselves and products they would develop.

5.2 Do monetary rewards have a different impact on software developers when solving more creative problems?

For the second research question, we wanted to determine if money had an impact on performance when different strategies were needed to solve the problem. We attempted to create two different tasks, one that required perceptual recognition and one that required a more creative solution using functional fixedness. Problems today are becoming more complex and require more creative solutions, and Glucksberg found that while financial incentives helped his participants in perceptual recognition tasks, it hurt those completing the functional fixedness task.

Our results showed that financial incentives had a negative impact on both problem-solving strategies in our tasks. P8 was the only one who had a faster time when he was offered money, completing the non-creative task. On average the non-creative task completion time was faster in the high and low drive groups compared to the creative task, but there was still an increase on average and results were not statistically significant between groups. P6 mentioned the money caused her to only focused on the "more complicated-looking part" of the non-creative task making her overlook a "very simple detail", and said she probably would have solved the task sooner without the monetary reward. The results for RQ2 shows that money did not have a different affect on debugging tasks using perceptual recognition or functional fixedness and we may need a better way to define 'creative' and 'non-creative' problems in software engineering.

5.3 Limitations

There were several limitations to our experiment. We hoped to get more people but were only able to get six participants to complete our study with three in each group. The subjects also weren't very representative of the population we want to study, using South Asian master's students to represent all software engineers. We also lost the recording for P7 and had one participant (P8) who is enrolled as a graduate student at NC State online and currently working in industry in Charlotte so we had to complete the study with him remotely using Google Hangouts.

During the study, several participants did not believe we were really offering money and thought it was fake. P5 specifically mentioned he didn't think the money was real until late in the task and he failed to solve it. We tried to divide the groups evenly based on the industry experience, but we assigned groups before the study when participants signed up for a time and two students (P2 and P3) did not show up for their session leading to a large difference in average years of work between Group 1 (2.6667) and Group 2 (6). Additionally, even though participants did not get a financial incentive to complete the study they did have some incentive by receiving academic credit.

6. CONCLUSION

Our study was unable to replicate the results from Sam Glucksberg's experiment on the influence of drive in different problem solving strategies for software engineers. He found that financial incentives negatively impacted the performance of subjects for a task that required functional fixedness and had a positive affect on participants in tasks that involved perceptual recognition. Even though our results were not statistically significant, it contributes insight into better methods to motivate software engineers which is becoming increasingly important as problems become more complex and require more creative solutions and our society becomes more reliant on software and technology.

7. ACKNOWLEDGMENTS

We would like to thank all of the students who participated in our study and thank Dr. Emerson Murphy-Hill who advised us on this project as part of the Software Engineering as a Human Activity course (CSC 710) and we also owe him $10.

8. REFERENCES

[1] R. Agarwal and T. Ferratt. Retention and the career motives of it professionals. In *ACM SIGCPR Conference*, pages 158–166, 2000.

[2] D. Ariely, U. Gneezy, G. Loewenstein, and N. Mazar. Large stakes and big mistakes. *Review of Economic Studies*, pages 451–469, 2009.

[3] J. Bloch and N. Gafter. *Java Puzzlers: Traps, Pitfalls, and Corner Cases*. Addison-Wesley Professional, 2005.

[4] J. L. Bowditch and A. F. Buono. *A Primer on Organizational Behavior*. John Wiley & Sons, inc., 4th edition, 1997.

[5] J. Burn, L. Ma, and E. N. Tye. Managing it professionals in a global environment. *SIGCPR Comput. Pers*, 16(3):11–19, 1995.

[6] D. Darcy and M. Ma. Exploring individual characteristics and programming performance: Implications for programmer selection. In *Proceedings of HICSS '05*, pages 314a–314a, 2005.

[7] K. Duncker and L. Lees. *On problem-solving*. Number v. 58 in Psychological monographs. The American psychological association, inc., 1945.

[8] A. Franca and F. da Silva. An empirical study on software engineers motivational factors. In *Empirical Software Engineering and Measurement, 2009. ESEM 2009*, pages 405–409. 3rd International Symposium on, 15-16 Oct 2009.

[9] S. Glucksberg. The influence of strength of drive on functional fixedness and perceptual recognition. *Journal of Experimental Psychology*, 63(1):36, 1962.

[10] A. J. Ko and B. A. Myers. Designing the whyline: A debugging interface for asking questions about program behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 151–158, New York, NY, USA, 2004. ACM.

[11] D. Pink. The puzzle of motivation. TEDGlobal, July 2009.

[12] S. L. Rynes, B. Gerhart, and K. A. Minette. The i mportance of pay in employee motivation: Discrepancies between what people say and what they do. *Human Resource Management*, 43(4):381–394, 2004.

[13] R. Sach, H. Sharp, and M. Petre. Continued involvement in software development: Motivational factors. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, pages 44:1–44:4, 2010.

[14] F. Tanner. On motivating engineers. In *Engineering Management Conference, IEMC '03*, pages 214–218, 2003.

[15] G. Tassey. *The Economic Impact of Inadequate Infrastructure for Software Testing*. National Institute Of Standards & Technology, May 2002.

[16] I. Uy. On the high turnover rate of software developers or how to retain your best software developers and programmers, 2010.

[17] V. H. Vroom. *Work and Motivation*. John Wiley & Sons, inc., 1964.

[18] K. Whitaker. Motivating and keeping software developers. *Computer*, 30(1):126–128, Jan 1997.

[19] V. Woods and R. van der Meulen. Gartner says worldwide it spending is forecast to grow 0.6 percent in 2016, 2016.

APPENDIX

A. CREATIVE TASK

```
1 students = list of all students in the system
2
3 /* Function to return the number of computer science graduate
4 students enrolled in the school.
5
6 Note: Undergraduate students taking classes towards the
7 Accelerated Bachelor's/Master's should also be included.
*/
8 GetCSCGradCount()
9    count = 0
10    for student = each element in students
11        add = false
12        if student.major == "CSC"
13            if student.degree in ["PhD", "MS", "MCS", "MA"]
14                add = true
15            else
16                for c = every class student has passed
17                    if c.subject=="CSC" and c.number>500
18                        add = true
19                    if add == true
20                        count = count++
21    return count
```

B. NON-CREATIVE TASK

```
1 /*Unity IDs must be 8 or more characters with at least one
2 letter followed by one or more numbers */
3 ValidateID(username)
4    Let letter, number = false
5    if username.length > 8
6        return false
7    for c = each character in username
8        if c is a letter and number == false
9            letter = true
10            continue
11        else if c is a digit
12            number = true
13            continue
14        else
15            return false
16    return letter and number
```