Penalty Policies in Professional Software Development Practice: A Multi-Method Field Study

Yi Wang
Department of Information Systems
City University of Hong Kong
Tat Chee Avenue 83#, Kowloon, Hong Kong
ywang1@acm.org

Min Zhang
Japan Advanced Institute of Science and Technology
923-1211, No. 1 Building 1-109
1-8 Asahidai, Nomi, Ishikawa
zhangmin@jaist.ac.jp

ABSTRACT

Organizational Punishment/Penalty is a pervasive phenomenon in many professional organizations. In some software development organizations, punishment measures have been adopted in an attempt to improve software developers' performance, reduce the software defects, and hence ensure software quality. It is unclear whether these measures are effective. This article presents the results of a multi-method field study that analyzes software engineers' perception towards penalty policies in relation to software quality in a software development process. The results were generated via both qualitative and quantitative methods. Through interviews, we collected the individuals' perception towards the penalty policy. By extracting data in a software configuration management system, we identified several patterns of defects change. We found that while a penalty mechanism does help to reduce software defects in daily coding activity, it fails in achieving programmers' maximum work potential. Meanwhile, experienced software programmers require less time to adapt to penalty policies and benefit from exist of less experienced developers. Some additional findings and implications are also discussed.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management - *Programming Teams, Productivity.*

K.6.1 [**Management of Computing and Information Systems**]: Project and People Management – *staffing, system development.*

General Terms

Management, Human Factors, Measurement.

Keywords

Penalty Policies, Software Defects, Perception and Performance of Software Developers.

1. INTRODUCTION

Software systems play an increasingly significant role in our lives. Unfortunately, software systems often fail to deliver as promised due to the existence of flaws and defects. Unidentified errors remain in many software systems in use today. Some software defects have received much attention, for example, South Korea's first rocket launch problem in Aug. 2009 [1], which reduced the availability of many important services. With the increase of software systems' complexity and the use of modern rapid development methodologies, avoiding, finding, and fixing software defects become even more difficult.

Many studies exist that focus on predicting, detecting, tracking and fixing techniques. Many existing software defect related techniques not only suffer from some kind of theoretical or methodological deficiency [13], but also ignore human and organizational factors in software development practice. Professional software development practice is a human-centered activity and largely is a social-technical process [10]. Software is created, maintained and used by human beings rather than machines. Organizational factors also influence quality of software systems significantly. Conway's law stating "Any organization that designs a system will inevitably produce a design whose structure is a copy of the organization's communication structure." reflects this point. Other empirical studies such as [16] also discuss the duality between product and organizational environment.

From an organizational or team perspective, what can help a software development organization avoid the occurrence of the software defects or identify defects as early as possible? This is a fundamental question that needs serious investigation. In this article, we enhance our understanding on this general area by focusing on organizational punishment. Punishment/penalty has been well examined but still has many different controversial opinions [e.g., 8, 9, and 22]. However, in software engineering domain, there is no formal literature that addresses this phenomenon. To bridge this research gap, we designed a study trying to empirically examine how organizational punishment or penalty works under the setting of real software development organizations and whether it helps to improve the performance of professional software developers (through reducing the software defects produced in daily coding practice).

Thus, we ask following research questions:

***RQ1:** What are the software developers' perceptions towards penalty policies? How do their perceptions changed with time?*

***RQ2:** Does penalty policies help to reduce the software defects and, to improve software engineers' performance? Do personal characteristics (e.g., experience) influence the effectiveness and patterns of how penalty policies work?*

This article reports on a field study of punishment/penalty in two online, multiplayer game development teams through a combination of qualitative research and quantitative research. This research goes beyond traditional face-to-face interviews by extracting useful information from the electronic repositories associated with the respective software development projects. This combination removes the common method bias of a single research method and brings more confidence in the completeness and soundness of our findings.

The rest of this article is organized as follows. Section 2 presents theoretical backgrounds of this research together with a brief introduction of related work. Section 4 states the major research settings. Section 5 discusses the methodology adopted in data collection and analysis. Section 6 summarized the major findings. Section 7 discusses some accidental findings, implications and limitations of this study. Section 8 concludes the article.

2. THEORITICAL BACKGROUNDS AND RELATED WORK

2.1 Organizational Punishment from Justice Perspective

There are many controversial opinions on the punishment towards employees. Conventional managerial wisdom often claims that punishments/penalty may lower employee morale, create negative influences to their performance, and influence the atmosphere of workplace resulting in side effects that outweigh any benefits [14]. Some studies report contradicting results that is some punishment can actually result in positive outcomes [3]. From an economics perspective, Sigmund et al suggest that punishing works much better than reward in promoting cooperative behaviors in a team context [6, 21].

In almost all studies that report positive effect of organizational punishment, "Fairness Heuristic" (or "*Justice Heuristic*" in some literatures) is the fundamental attribute of the positive outcome. In [4, 5, and 22], they proved the possibility that some positive organizational effects would be generated if the introduced punishment/penalty mechanism is fair "enough" because the punishment/penalty mechanism is easily accepted by the employees.

Another important topic in organization punishment studies is the third person effect, which refers to the observation that punishing one individual in a team will also influence other people's behaviors. This phenomenon is well described by a traditional Chinese proverb: "*Punish a chicken as a warning to monkeys.*"

2.2 Human's Self Motivation on *Threats to Punishment*

Human beings incline to avoid potential punishment. In Deci's [12] highly-cited paper, he claimed the threats of punishment for poor performance would extrinsically motivate individuals to improve their performance although it would also lead to some decrease in people's intrinsic motivations. In [11], it is further pointed out that the use of promised rewards or threatened punishment is a ubiquitous motivational strategy, especially for the uninteresting tasks. In [18], McGregor *et al* point out that punishments commonly help to motivate employees usually with management by direction and control. As an important aspect in human management, performance management also addresses the usefulness of punishment in improving professional employees work performance. Empirical studies n software engineering [20] have demonstrated that fear of punishment is an important motivating factor for developer's behaviors.

2.3 Punishment in Software Development

In Information Technology (IT)-intensive, task-oriented teams, punishment is also an important factor that influences team productivity significantly. In [24], they presented punishment as an important dimension of the task pressure, which will influence their subjects' group performance. In [15], having a clear standard for reward and punishment is an essential part of systematic management control for software development team. Patton and Jayaswal [19] also claimed that proper reward/punishment system will help to build trustworthy software systems.

Our work is different from previous studies in two aspects: (1) we provide case studies in a real software development environment rather than discuss without empirical evidence as support, and (2) we employ both qualitative and quantitative methods in our case studies.

3. CONCEPTUAL FRAMEWORK AND PROPOSITIONS FOR OUR CASE STUDIES

To frame our studies, we make use of a three-element conceptual model depicted in figure 1, which is based on the major research questions. The three elements are: (1) punishment/penalty policy, (2) perception of software development engineers (SDEs), and (3) influence to program quality. Each element contains sub-elements. This framework is proposed under the rationale that punishment/penalty policy will have significant impact on the perception of SDEs, hence alter their on-job behaviors, and introduce influence to the program quality.

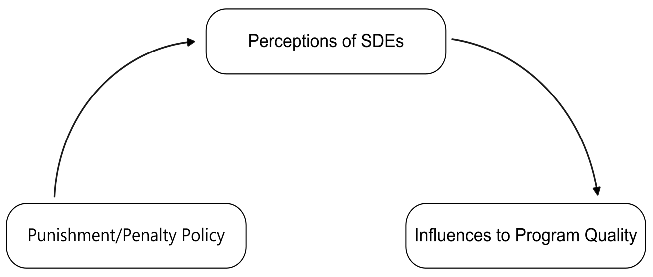

Figure 1. The Conceptual Framework.

Based on above theoretical backgrounds mentioned in section 2 and empirical research results, we hypothesize:

Proposition 1. *("Justice Penalty Works")* The justice penalty mechanism will help to reduce software defects positively.

Proposition 2. *("Fairness Is the Most Important Thing")* For software developers, they are concerned about the fairness of the penalty policies. In particular, the penalty policy should avoid being perceived as a measure to reduce a developers' income.

Proposition 3. *("Third Person Effect")* If somebody in the team were punished, others' concerns would increase; and their code quality would also have some improvements.

Proposition 4. *("Warm up Period is Necessary")* The warm up period will help the software developers better adjust to penalty policies.

In addition, considering the complexities of software development, experience may also influence software developers' capability; hence, we have the following additional propositions:

Propositions 5. *("Experience Helps")* The experienced software developers will need less time to adjust themselves in adapting to penalty policies.

Furthermore, it is natural to ask whether the penalty policies would cause a significant increase of software developers' workload. For example, with the existence of penalty policies, software developers tend to spend more time on testing and debugging, it seems their workload may increase. However, exist software engineering research results conclude that, finding and fixing software defects earlier saves more effort than fixing them in later development and maintenance phases.

Propositions 6. *("No Extra Workload")* Software engineers will not experience significant workload increase after introducing penalty policies.

4. ORGANIZATIONAL SETTINGS

4.1 Alpha and Its Game Development Department

This research was conducted in the multiplayer, online game development department of a China major online entertainment provider Alpha[1]. Alpha is one of the top 3 multiplayer, online game service providers in mainland China market. The number of concurrent online users often exceeds 10,000,000 at its peak (often between 9:00pm and 01:00 am in GMT +8:00).

The games running on Alpha's platforms are from two sources. Some are developed by the third-party game designers, e.g., Softstar, EA sports, while others are developed by Alpha's internal development department. Alpha's internal development department has over 1000 full-time software engineers focusing on multiplayer, online game development. In the rest of this article, we use "*Beta*" to refer this internal department.

In Beta, software engineers are organized into development teams according to different products. A typical team contains around 10 developers. In addition, there are other kinds of teams focused on product quality assurance, which consist primarily of the testing specialists and quality assurance engineers. These quality assurance teams are independent to development teams under different working process.

4.2 Workflow of Development Team

In daily development practice, software engineers are assigned a small piece of development task, e.g., writing a program to find players physical location or assigning game equipment to players randomly. For each finished task, source code must be submitted through the software configuration management (SCM) tool (In Alpha, Rational ClearCase is the major SCM tool). Only the program manager (PM) and senior SDE have permission to create an entry in the Rational ClearCase[2] for each assigned task.

The developers are asked to conduct unit test on their code before submission. After their own testing, the quality assurance team (independent to the development team) will examine the code, if there are some defects, or some other problems in the code (e.g., incomplete documentation, bad coding style, etc), and will send a report back to the code author through the SCM system for revisions until the submission is accepted. If submitted code can not pass the quality assurance examination before a pre-set submission deadline, it is recorded as a fail submission. This process is managed and monitored by the Rational development platform.

4.3 Detailed Penalty Policy

The penalty policy was launched in Sep. 2008. In the first two weeks, the PM only recorded unsuccessful submissions but not enforce the monetary fines. The real punishment mechanism started after the China's National Day vocation.

Followings are the penalty rules implemented by Beta:

1: If a programmer has twois tracked twice unsuccessful submissions in a week, this programmer will be fined CNY100 (around $14.70).

2: For each unsuccessful submissions beyond the initial tow unsuccessful submissions in a single week, the programmer will be finned and additional 100 RMB. 3: The maxim penalty is CNY800 (around $120) per month for each individual and CNY3000 (around $450) for the whole team in a financial month.

4: The money collected through penalty will be used for team activities, every team member has right to monitor the use of collected penalty sum.

5. Penalized persons will remain anonymous. If some person is penalized, other members only know that someone has been penalized (via email), but not who is penalized.

Besides rules stated above, Beta also provided a "match" fund for team activities. For example, if team Omega collected CNY1000

[1] For privacy consideration, we use Alpha, Beta and Omega to mention the real name of the subjected company, department and team.

[2] Rational ClearCase: http://www.ibm.com/software/awdtools/clearcase.

penal sum, Beta will provide equal sum of money to team Omega for their team activities.

Obviously, the penalty mechanism adopted in Beta is impartial. The penalty decision is totally based on the evaluations executed by the independent third party.

4.4 The Implementation Process of the Penalty Policy

The penalty policy started from 15/09/2008. From 15/09/2008, two weeks were selected as the "Warm-Up" period to make software engineers familiar with the penalty policy. During this 2-week "warm-up period", if a project manager found one of his/her subordinate should be fined; he or she would issue a "*bill*" via email. However, people who received a "*bill*" did not need to pay.

The real penalty policy started at 08/10/2008 after 7-days national day vocation (01/10/2008-07/10/2008). Till we wrote this paper, this penalty policy was still working in Beta.

5. METHODOLOGY

We combined qualitative and quantitative research method in this research to achieve both acceptable depth and breadth. As far as our current knowledge, there is no established theory or empirical analysis that addressed the organizational punishment/penalty in professional software development process. That is, we were in lack of conceptual frameworks to guide the research efforts in this field. Besides, we were not sure whether the existing research on organizational punishment could be directly applied to under the unique setting of software development. Meanwhile, there are also no other benchmarks for us to follow and compare. Based on these reasons, we first used qualitative methods to get an intuitional understanding of our findings, and then, verified some of them quantitatively. This multi-methodology ensures the reliability of this research.

5.1 Qualitative Methods

During our field study in Beta, the researchers conducted a series of semi-controlled face-to-face interviews with the software developers in two teams. 10 persons were selected as interviewees.

Story Telling for Individual Data Collection

As part of whole study, we developed a story telling approach to collect software developers' individual perceptions towards penalty policies. The story telling approach was adopted as an integrated part of whole interview process. The interviewees were asked to freely tell their personal stories at the beginning of each interview. The following interview questions were based on their narratives.

However, qualitative method cannot provide comprehensive and reliable answers to both research questions, especially RQ2. To gain better understanding on the relationships between the penalty mechanism and software engineers' performance, we also conducted quantitative analysis based on data extracted from Beta's software configuration system.

5.2 Quantitative Methods

To further verify the findings that emerged from the interviews and field studies, the researchers used information stored in corresponding software configuration management (SCM) systems. As we mentioned before, Beta used IBM Rational ClearCase as it major SCM tool, therefore, we retrieved data in ClearCase to get the required information. According to Beta's policy, each submission's information was kept in SCM system until the close of the software product line. For each submission made by the software engineer, we examined the reports from the quality assurance team. In these reports, software defects found by QC team were recorded.

We selected the first 4 months (form 09/2008 to 12/2008, we collected information in 09/2008 as the benchmark for comparison) log records. This is due to two reasons. In the first place, to fulfill the information disclosure policy of Alpha, we could not retrieve the working data from the last 6 months. Secondly, considering that the software developers became familiar with the penalty policy, the influence of penalty policies become stable and well understood. Another consideration was that there were no new programmers added to the study teams during this period, which provided some convenience to our study.

After we retrieved the needed information from ClearCase system, we examined and recoded it to make it more suitable for statistical analysis. After the data re-codification, we secured 21 cases of software developers' data in 82 working days.

For each data case, variables about personal and work information were extracted. Five variables were introduced as follows.

5.2.1 Personal Information

1. Education Level

The education levels were coded into three categories: "non-university level education", "university level education" and "postgraduate level".

2. Experience

We categorized all subjected software engineers into two groups: experienced and novice developer. If the developer had less than one year development experience (till 09/2008), we treated them as novices. In fact, most novice developers had less than 2 months full-time development experience[3].

3. Gender

In our study, male was coded as "*1*", while female was coded as "*0*". This information was collected to verify whether gender influence the pattern of software engineers' work performance.

5.2.2 Recorded Work History

For each day's work history, we extracted two kinds of information.

4. Number of Detected Defects (NDD)

For each studied software developers, we extracted the sum of the defects found by the quality assurance team each day. Through collecting this data, we could build patterns for the change of defects from both individual and group perspective. This can also be used to verify other propositions we proposed.

[3] This is not just a coincidence. Alpha hires many new fresh campus graduates in 2008 to fulfill the need of its repaid expansion, their appointments usually started at July, 2008.

5. Time Stamp for Each Penalty (PTM)

The time stamp of each penalty was selected as a variable. According to our data set, there were 31 penalties that occurred during the survey period. The granularity for each time stamp was day. This variable was mainly used to test the 3rd person effect.

Besides the information mentioned above, we also collected some additional data. Obviously, the collected data is longitudinal data. The major focus of this study was determining a pattern of the software quality change attributable to the penalty policy. Curve estimation [2] approach is quite suitable for this study. Many curve estimation methods have been increasingly used in sociology and anthropology research but not widely adopted in empirical software engineering research domain. Given the small size of data cases in this study, we adopted the simple several curve estimation models (i.e., quadratic, cubic, growth, etc). This technique can capture patterns and discover causes of variability in patterns at the same time, making it a suitable tool for researchers who are interested in identifying the antecedents of certain outcomes at a given time point, as well as the determinants of changes in patterns over time. Through using a set of repeated observed measures, it can help researchers identify the pattern of changes over time. SPSS16's integrated curve estimation tool was adopted to check whether there are some patterns we wanted. We also conducted some regression check after we got preliminary result and binding different models for different phases.

In this study, we focused on the change patterns of the number of detected defects. Through study the change patterns of different employee groups, we hoped to verify the proposed propositions, together with any possible meaningful findings.

5.3 Demographics

The study was conducted in two development teams. The first team has 13 members while the second one has 14 members. In each team, there were some individuals who served as project managers and team support staff. The total number of code contributors was 21. All of them had college level or above education, 7 obtained post-graduate degree or equivalent advanced degrees. According to the criteria set for experience, 12 were experienced while 9 were novice developers. The average experience was 1.59 years, with the standard deviation of 1.84. The two teams mainly consisted of young men less than 35 years old (18 males, 3 females).

6. FINDINGS

6.1 Software Engineers' Perceptions

6.1.1 Changes of Overall Concerns

According to collected interview results, we found surveyed interviewees concerns towards the penalty policy decreased as time went on. In first month of penalty policy introduction, people paid excessive attention to penalty issues. As a software engineer reported: "*In the first several weeks, I and my colleagues were all quite worried about this. It was the hottest topic in our lunch discussions. But this over reaction did not last long. After several weeks, we only talked it when we informed there was some person was fined.*" He further illustrated the reason what contributed to such a change, *"Compared with our salary, the penalty is not much; one person could be fined at most CNY800 in a month, less than 10%, so we do not need to care too much. Besides, we just need to pay small amount extra attentions in writing code to avoid most fail submissions!"*

The decrease of concerns is also due to software engineers finally finding that it is not quite difficult to avoid penalty. Another interviewee pointed out: "*Most problems were not caused by technical difficulties, but our carelessness. If we pay a little bit more attention, they can be avoided.*"

There were several persons who showed more concerns than others. However, these are mainly results from their individual characteristics but not the penalty policy itself.

6.1.2 If Penalty is Fair, It Is OK

Most of employees admitted that they could only accept a fair punishment/penalty policy that does not intend to treat some staff differently. In this case, penalty policies based on the third party's (Quality Assurance Team) report are no doubt a fair penalty.

In fact, fairness is not only important in penalty design, but also essential in nearly every aspect in organization setting [17]. It is essential not only for punishment/penalty policy design, but also for staff evaluation and reward system design.

6.1.3 Upper Penalty Limit is Necessary

Besides fairness issue, setting upper penalty limit is necessary in avoiding upsetting employees'. According to most interviewees, this policy ensures their basic salary was not to be influenced significantly by the penalty policy. As an interviewee commented, "*once I know there was an upper limit, I felt a little bit better. If no such limit, I think most of us would strike for protest.*"

6.1.4 No Significant Workload Increase Reported

As we predicted in proposition 5, introducing penalty policies did not cause significant increase on workload. An interesting phenomenon like this follows. When we asked questions about "workload", the interviewees typically said *"I need to conduct more testing and debugging, pay extra attentions in writing code. But, I spend less time in revising my programs after submitting it to the system."* Obviously, increased effort on testing can be compensated by less later efforts on further program revision and maintenance.

6.1.5 Increased Team Cohesion

Conventional opinions often declare that punishment/penalty would damage the team atmosphere. However, our study provided conflicted results. It is natural to ask the reasons for this. Based on the interviews, the most plausible explanation is the use of collected penalty sum. According to the penalty rules, collected money (together with the match fund provide by Beta) would be used for team activities. Obviously, regular team activities help to increase team cohesion. The interviewees' feedbacks also reflect the team activities make them know other members better.

6.2 Software Engineers' Performance

In this sub-section, we will discuss the major findings based the quantitative results. We first attempted to compute the overall change pattern of the total found software defects. Based on the collected data, we get two curves to describe the change patterns of number of defects (per day) produced by the novice and experienced developers respectively. Figure 2 shows the detailed information.

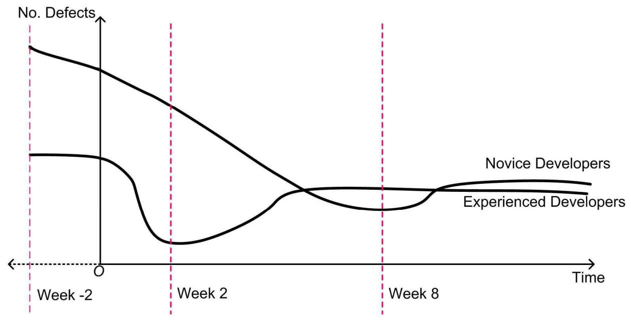

Figure 2. The Estimated Curve for Two Group of Developers (Novice and Experienced).

6.2.1 Penalty Works, But Not so Significant as Expected

In the first place, both two curves show that significant decrease trend of the number of detected defects per day. However, we can find an interesting "U" shape for both curves. So, what is the meaning of the "U" shape pattern?

This question is really simple. The "U" shape pattern indicates that this penalty policy fails to utilize the maximal potential of the developers. For developers, they first tried their best to avoid being fined. However, after a short period, they found they did not need to pay so many efforts to avoid being penalized. Therefore, the defects number began to increase until stable. Therefore, the policy makers' may be disappointed for the failure of utilizing developers' greatest potential. However, the penalty policies did successfully reduce the number of detected defects significantly.

6.2.2 Warm-Up Period Is Not Very Useful

This finding contradicts to the proposition 4 we presented in section 3. Leaving a short period for warming up is a common managerial practice when introducing new policy. However, our result show this period is useless, especially for experienced programmers. This may result the fact that the "mock" penalty will not draw enough developer's attention. To verify this point, further studies are needed.

6.2.3 Third-Person Effects

As we mentioned before, we collected time stamps for each penalty. To verify whether of the Third-Person effects exist, we conducted following computation to recode data:

- Step 1: Count the sum of all reported defects (except the defects produced by the fined person) in the day when somebody was fined,
- Step 2: Count the sum of all reported defects (except the defects produced by the penalized individual) the day immediately after the day somebody was fined,
- Step 3: Add another category variable to differentiate these two groups of data.

For the 31 cases of penalty execution during the observation period, we get another 62 pairs of data. Then we used ANOVA test to find whether there are significant difference. The results are (*F: 41.462, P-Value: .000*).

6.2.4 Experience Helps

We could admit that experienced software developers used less time to adjust themselves to the new policy. From figure 2, we can find experienced software developer's quality assurance failure curve achieved stability quicker than the novice developers.

We computed the time before the quality assurance failure curve becomes stable for each case, recorded them, and then conducted one-way ANOVA test to verify this point, which also indicates significant short adaptation period for experienced developers.

The tails of these two curves suggest that experienced developers benefit more due to the existing of novice developers. It is surprising that they adopted the most economic strategy to protect their own interest, which is just performing a little better than the novice ones. We have consulted several subjects to make sure whether they intended to do this. Most of them refused to admit they use this strategy with intention. Their choices are more intuitive; they just followed their feelings on the amount of efforts that make them free of penalty. This reflects the experienced experience can make precise judgment on other team members' capability in their subconscious.

Table 1. Summary of Major findings

No.	Findings	RQ	Prospostion	Qualitative/Quantitative
1	Concerns Decrease	RQ1	n.a.	Qualitative
2	Importance of Fairness	RQ1	P2*	Qualitative
3	Upper Penalty Limit is Necessary	RQ1	n.a.	Qualitative
4	NO Significant Workload Increase Reported	RQ1	P6*	Qualitative
5	Team Cohesion Increase	n.a.	n.a.	Qualitative
6	Penalty Works, BUT Not so Significant as Expected	RQ2	P1*	Quantitative
7	Warm-Up Period Is NOT Very Useful	n.a	P4**	Quantitative
8	Third-Person Effects Do Exist	RQ2	P3*	Quantitative
9	Experience Helps	RQ2	P5*	Quantitative

*Notes: * Supported, ** Rejected*

6.3 Summary of Findings

We use table 1 to summarize above findings with corresponding research questions, propositions, and research methods.

7. DISCUSSIONS AND IMPLICATIONS

7.1 "*SIDE*" Findings

Besides the findings mentioned in prior section, there are still several interesting *"side"* findings.

7.1.1 Increased "Discontentment" towards Quality Assurance Team

An interesting phenomenon discovered during our interviews is the increase of "discontentment" towards Quality Assurance Team. In several interviewees' descriptions, people in Quality Assurance Team are bad guys, who always try to make them be fined. One interviewee's words represent this negative attitude: "*These bad guys do not write any useful code, but always finding pleasures through making us be penalized. Even a small fault such as spelling mistake in comment, they also put it into report.*" This is reasonable from the "***common enemy***" perspective.

7.1.2 Some Persons Are Really "Unfortunate"

In studied period, 31 cases of penalty were recorded. However, the distribution of penalty is not even for each individual. 2 persons were penalized more than others; they were fined 12 times, nearly 40% of all penalties. This is not accidental for two "Unfortunate" guys. This indicates these two developers may not capable (or suitable) for their role. In fact, one of them was fired in January. 2009 for bad peer evaluations. To some extend, the penalty may be potential for staff evaluation. However, the organizations should be very careful to use it as a criterion in developer evaluation because it can not reflect comprehensive capability of staff and is also easy to be influenced by many other factors (e.g. task attributes, etc.)

7.1.3 Is Gender or Educational Level an Important Factor?

According to some impression we acquired in the story telling session, we found female developers expressed more complaints at the beginning stage of penalty policy. Therefore, it may take longer time for the female developers to adjust themselves.

However, we could not verify this point through statistical measures. In our data set, there are 3 female cases. The number of cases is too small to be statistical significant. Besides, two of them are inexperienced. This point is not conclusive but needs further investigation.

For the educational levels, the situation is similar as gender factor. In general, university-level education does not have great differences than postgraduate-level education. For professional software development, accumulating experience is more important than earning high educational degrees.

7.2 Which is more important for Novice Developers, Penalty or Experience Increase?

It is safe to conclude that penalty mechanisms do help the experienced software developers improve their performance. However, we still face a question on whether novice developer's performance improvements are also caused by penalty policies or just the result from experience accumulations. From figure 1, the performance curve's initial part does suggest experience increases do help improve novice developers' performance.

To answer this question, we need a close analysis towards the novice developers' performance curve. First, if we suppose penalty policies have nothing with novice developers' improvements, so what should the performance curve's shape look like? The answer is quite clear. The curve should show a keeping decrease trend and without any wave trough (NO "U" shape pattern). However, we can see an apparent wave trough in the eighth week. Based on this, we can draw a conclusion that novice developers' performance improvements are also partly caused by the penalty policies.

7.3 Promotion vs. Prevention

In organizational level, there are two kinds of measures to motivate employees. The first one is promotion while the second one is prevention. A promotion focus, would involve a state of eagerness to attain advancement and gains whereas a prevention focus would involve a state of vigilance to assure safety and non-losses. In another word, if employers want to facilitate their employees to utilize their creativity to solve problems, promotion measures will be a wise choice. But, if employers want to prevent their employees to do some "wrong" things, prevention measures could be adopted.

The promotion and prevention perspective provides a theoretical lens for us to understand why penalty policy in Beta showed some effectiveness. Now, we can have a close look about the case described in this article. Figure 3 depicts a typical online multiplayer game development process which contains 7 major sub-processes.

From this figure, we can easily identify the sub-process need high creative thinking is the first four. The programmers' work most concentrates on following two sub-processes. These two processes do not ask programmers to be creative or imaginative. They just need to follow specifications to finished coding works. According to our observation in Beta, people involving in first four sub-processes are often awarded to promote their creativity, while programmers are often fined to prevent their careless mistakes.

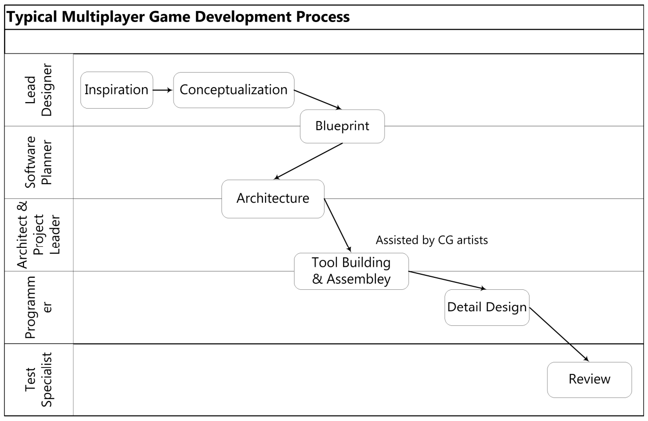

Figure 3. Typical Online Multiplayer Game Development Process and People Involving in Each Sub-process

7.4 Implications for Future Practice

7.3.1 Punishment/Penalty Policy Design

As we stated before, there are many conflicting opinions towards launching a penalty policy in organizations. It is also hard to give an explicit answer whether organizational punishment or penalty can contribute a positive influence to the software development practice. However, we could prove that the success of a punishment/penalty policy depends on various contextual and subjective factors whether from the people's perception or objective measurement perspective. It is worth both practitioners and academic researchers pay more efforts on this topic.

Besides, the penalty practice in Beta at least suggests a possible way on how to implement penalty policies in typical software development organizations. We summarize following 3 guidelines for launching penalty policy in software development:

Guideline 1: Penalty policies should be well designed to ensure the justice (In this case, the independent third party takes charge of the evaluation, hence avoids the conflicts of interests).

Guideline 2: Penalty policies should be clearly delivered with the information that the intention of penalty policy is not trying to cut cost through lower the employees' net income but to motivating them to deliver high quality work.

Guideline 3: The last but not the least point is the penalty sum should have an upper-bound, e.g. 5% of month salary, for each individual. Without the upper-bound, software engineers may feel upset and refuse to accept it.

It is also important to remember that no policy can make every stakeholder happy; therefore, achieving compromise is more important than forcing developers to obey.

7.5 Threats to Validity

7.4.1 Internal Validity

From an internal validity point of view, the data collection was conducted by one researcher. For a single project, the data collected by the same person under the same protocols. There was no necessity and possibility for the researchers to influence the subjects' story telling in narratives collection. All the software engineers and their managers who engaged in this study have no conflict of interests with the researchers. The reasons and impacts were self-reported in free form to remove any bias that could have been introduced by the researchers influence. Further more, the researchers had no privacy relationship with the research subjects, formal agreement and fully acknowledgements to sensitive information are achieved.

7.4.2 External Validity

From an external validity point of view, this study is based on the software development organizations locating in China. Although we made this decision mainly because of the convenience considerations, it is still an appropriate choice, for Shenzhen is one of the cities where have highest developed knowledge intensive (especially game design) industry. Besides, Shenzhen is also a city high in diversity. It is possible that the results will be still valid in other areas in China. However, for the success of software development depends on a potentially large number of relevant context variables including the organizational factors and task specific factors such as domain knowledge and task familiarity. We can not ensure the results still work in other settings. For this reason, we cannot assume a priori that the results of our study generalize beyond the setting in Alpha. Researchers would become more confident in a theory when similar findings emerge in different contexts [7]. Towards this end we intend that our case study will be replicated in different setting by the researchers (ourselves are also included). This research also can be treated as the benchmark for other future similar research conducted under other settings.

7.6 Limitations

The major limitation of this study is the relative small sample size (total 21 data cases). Due to the policy of sensitive information disclose, we can not access more data from Alpha. The interview subject selection process is also not random. Besides, we also can not ensure the results of this study still work well in other settings. However, this is also another aspect of this study's possible value; our research can be treated as the benchmark for other future similar research conducted under other different organizational contexts.

Another limitation is that our study is still far from mature. With more data or experience with this topic, other related issues may be apparent. For now, we think it is more important to consider how well this research supports the future practices and researches of punishment/penalty in software development organizations. In particular, does it help to:

- Understand the influence of punishment/penalty in software development organizations better,
- Explore new space and directions for research on this issue,
- Draw attentions from both practitioners and researches,
- Provide useful implications to penalty policy design and future practices.

Our study has great potential in this regard. We still hope to identify new issues to broaden and enhance our knowledge on this topic.

8. Concluding Remarks

Organizational punishment is an issue full of different opinions. Some of them even totally oppose to some others. For organizational punishments in software development, few formal works address this topic. If no empirical evidence for this pervasive phenomenon, little progress can be made on leveraging organizational factors to improve software systems' quality. To fill this gap, we investigated the punishment/penalty in software development organizations. Rather than only focusing on the developers perceptions towards penalty policy, we also examined the penalty policies influence to program quality.

In general terms, our study suggested that, at least in Beta, the penalty policies have gained some success in motivating software developers deliver better programs although the potential of software developers is still not fully utilized. Through the interviews and quantitative data analysis, some interesting phenomena have been discovered, such as the differences between experienced and novice developer, and third person effects. The implications of this study are also discussed. We hope this study can be treated as the benchmark for the future replication and extension researches in different contexts and settings. Some other research methods such as organizational experiments are also worth to try.

9. ACKNOWLEDGMENTS

The authors want to show their thanks to all participants of this study, especially for several project mangers for helping us to get an opportunity to collected data from their team. We also want to thank Dr. Bolloju for his comments to this work.

10. REFERENCES

[1] Channelnewsasia: http://www.channelnewsasia.com/stories/afp_asiapacific/view/449967/1/.html, accessed at 14th Sep. 2009.

[2] Introduction to Curve Estimation: http://faculty.chass.ncsu.edu/garson/PA765/curve.htm, accessed at 14th Sep. 2009

[3] Arvey, R. D., Davis, G. A., and Nelson, S. M. 1984 Use of discipline in an organization: A field study. Journal of Applied Psychology, 69(3): 448-460.

[4] Ball, G.A., Treviño, L.K. and Sims, H.P. Jr. 1994 Just and unjust punishment incidents: Influences on subordinate performance and citizenship. Academy of Management Journal, 37 (2): 299-322.

[5] Ball, G. A. and Sims, H. P. 1991 A conceptual analysis of cognition and affect in organizational punishment, Human Resource Management Review, Vol. 1, 227-243.

[6] Baranski, B et al. 2006 High-order punishment and the evolution of cooperation. In Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO2006). ACM, 379-380.

[7] Basili, V., Shull, F., and Lanubile, F. 1999. Building Knowledge through Families of Experiments. *IEEE Transactions on Software Engineering*, 25(4), pp. 456-473.

[8] Butterfield, K., Treviño, L.K. & Ball, G.A. 1996. Punishment from the manager's perspective: A grounded investigation and inductive model. Academy of Management Journal, 39, pp. 1479-1512.

[9] Butterfield, K. D., Trevino, L. K., Wade, K. J. and Ball, G. A., 2005 Organizational punishment form the manager’s perspective: An exploratory study. Journal of Managerial Issues, Vol. 17, Issue 3, pp. 363-382.

[10] Cataldo, M., Herbsleb, J.D., Carley, K. Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity, 2nd Symposium of Empirical Software Engineering and Measurement, Kaiserslautern, Germany, 2008.

[11] Deci, E. L. The effects of contingent and noncontingent rewards and controls on intrinsic motivation. Organizational Behavior and Human Performance, Vol. 8, pp. 217-229, 1972.

[12] Deci, E. L. and Ryan, R. M. Intrinsic motivation and self-determination in human behavior, 1985.

[13] Fenton, N. and Neil, M. 1999 A critique of software defect prediction models, IEEE Transaction on Software Engineering, Vol. 25, No. 3, pp. 1-15.

[14] Luthans, F., and Kreitner, R. Organizational behavior modification and beyond. Glen-view, IL: Scott, Foresman, 1985.

[15] Keskin, H. 2009 Antecedents and consequences of team memory in software development projects. Information and management 46 (2009), pp.388-396.

[16] MacCormack, A., John Rusnak, and Carliss Y. Baldwin. 2008 The Impact of Component Modularity on Design Evolution: Evidence from the Software Industry. Harvard Business School working papers. http://hbswk.hbs.edu/item/5831.html. accessed at 18/09/2009.

[17] Martin, C. L., and Bies, R. J. 1991 Just laid off, but still a "good citizen"? Only if the process is fair. Paper presented at the annual meeting of the Academy of Management, Miami Beach, FL.

[18] McGregor, D. and Cutcher-Gershenfeld, J. 2006. The Human Side of Enterprise, McGraw-Hill Skarlicki, D. P. and Kulik, C. T. Third Party Reaction to Employee (mis)Treatment: An Justice Perspective. Research in Organizational Behavior: An Annual Series of Analytical Essays and Critical Review. Vol 26. Ed. by and Barry M. Staw, Roderick Moreland Kramer. pp. 183-230. Elsevier 2005.

[19] Patton P., and Jayaswal B. 2006 Design for Trustworthy Software: Tools, Techniques, and Methodology of Developing Robust Software. Prentice Hall.

[20] Sharp,H., N. Baddoo, S. Beecham, T. Hall, and H.Robinson, 2009 Models of Motivation in Software Engineering, Information and Software Technology 51, 2009, pp. 219-233.

[21] Sigmund, K., C. Hauert, and M. A. Nowak. Reward and punishment. In Proceedings of the National Academy of Sciences of the United States of America, volume 98, pp. 10757-10762, 2001.

[22] Treviño, L.K. The social effects of punishment: A justice perspective, Academy of Management Review, 17: 647-676, 1992.

[23] Treviño, L. K., & Weaver, G. R. 1998. Punishment in organizations; descriptive and normative perspectives. In M. Schminke (Ed.). Managerial ethics: Moral management of people and processes: 99-114. Mahwah, NJ: Lawrence Erlbaum.

[24] Vance Wilson, E., and James R. Connolly. Effects of group task pressure on perceptions of email and face-to-face communication effectiveness. In Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work, pp. 270-278.