The Effort Distribution Pattern Analysis of Three Types of Software Quality Assurance Activities and Process Implication: an Empirical Study

Qi Li
University of Southern California
941 w. 37th Place Los Angeles, CA
90089-0781
qli1@usc.edu

ABSTRACT
Review, process audit, and testing are three main Quality Assurances activities during software development life cycle. They complement each other to examine work products for defects and improvement opportunities to the largest extent. Understanding the effort distribution and inter-correlation among them will facilitate software organization project planning, improve the software quality within the budget and schedule and make continuous process improvement. This paper reports some empirical findings of effort distribution pattern of the three types of QA from a serial of incremental projects in China. The result of the study gives us some implications on how to identify which type of QA activity is insufficient while others might be overdone, and how to balance the effort allocation and planning for future projects, improve the weak part of QA activities and finally improve the whole process effectiveness under the specific organization context.

1. INTRODUCTION
For a successful project, acceptable quality must be achieved within an acceptable cost and a reasonable schedule. Generally, if a high-quality product is required, more effort would be put on quality assurance activities to achieve the required quality goal. IEEE's definition of Software Quality Assurance is: "A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements and a set of activities designed to evaluate the process by which the products are developed or manufactured"[1]. Review, process audit and testing are three popular quality assurance activities during software life cycle. Review, often referred to requirements review, formal design reviews, peer reviews (inspection and walkthrough), is usually considered to be the most effective quality assurance activity and the best industry practice according to many researches[2-5]. The maximum benefits come from review would prevent defects and save fixing time in future work [6]. However, some experts also suggest not expect to replace testing with reviews [7]. The two quality assurance activities capture different types of defects. Reviewers can't tell how fast and reliable the system will be, how user-friendly the system is by just reviewing and deducing from the abstract requirement, design documents or codes. Testing serves as a hands-on experience of the actual and operational system that reviews can't achieve by imaging [7]. Although process audit has an indirect impact on the software product, but it has been widely accepted that [8] process audit can help developing team to work on the right direction, adopt and tailor the most effective method that comes from the best industrial practice to the specific context of the project, improve the process effectiveness and finally improve the software quality.

Since the three types of quality assurance activities are complementary and can't be replaced by each other to achieve the quality goal, how to balance the effort allocation among them to improve the process and resource allocation efficiency and maximize the ROI of the entire quality assurance if we consider them as an investment to the software development life cycle is still an important yet challenging issue for project planning. In this paper, based on the data from a Chinese software organization, we do an effort distribution pattern analysis of the three types of software quality assurance activities and the result gives us some implications on how to identify which type of QA activity is insufficient while others might be overdone, and how to balance the effort allocation and planning for future projects, improve the weak part of QA activities and finally improve the whole process effectiveness under the specific organization context.

The remainder of the paper is structured as follows: section 2 will introduce the objective, subject, and approach of the study; section 3 will discuss the major results of the activity effort data analysis and process implication; section 4 is the conclusion.

2. OBJECTIVE, BACKGROUND AND DATA SOURCE OF THE STUDY

2.1 Objective of the study
The work performed in this study aims to supplement the sparse work on empirical assessment of effort distribution patterns of quality assurance activities, develop understanding to the effects of different types of quality assurance activities during software life cycle, and provide guidelines and process implication for quality managers in planning and balancing the effort distribution to maximize the ROI of QA activities.

2.2 Background and data source
The empirical study is based on the data from a series of incremental software projects within a medium-sized software organization in China. The lead product is the software quality management platform (SoftPM) platform, targeting to facilitate the process improvement initiatives in many small and medium software organizations in China. The quality management group is responsible for implementing the three types of QA activities during the whole project life cycle. Every version of this project share the same quality goal before releasing: that is test cases coverage rate is 100%, all planned test cases executed and passed; no non-trivial defects are detected during at least one day of

continuous testing, and satisfy the quality goal of 0.2 defects/KLOC when released.

We extract raw data from related plans, records and reports, consolidate the original information, resolve the inconsistency and finally get five versions QA data as shown from Table 1 to Table 3. And the meanings of data items are explained in the italic words below the table.

Table 1. Effort and its Percentages of Each QA Activity

Hrs (%)	E_Re_Req	E_Re_Des	E_Re_Cod	E_Re
V2.5	269.7(3.29%)	132(1.61%)	NA	401.7(4.90%)
V2.6	24(2.77%)	17(1.96%)	NA	41(4.74%)
V3.0	41.5(0.13%)	79(0.25%)	NA	120.5(0.38%)
V3.1	458(2.82%)	13(0.08%)	135(0.83%)	606(3.74%)
V3.1.1	35(2.14%)	74(4.52%)	18(1.10%)	127(7.76%)

Hrs (%)	E_PA	E_Test	E_Total
V2.5	146(1.78%)	2587(31.58%)	8192
V2.6	21.7(2.51%)	187(21.61%)	865.5
V3.0	472.65(1.49%)	15466(48.81%)	31684.5
V3.1	162(1.00%)	7787.6(48.03%)	16215.1
V3.1.1	NA	365.5(22.33%)	1636.5

E_Re_Req: Review effort in Requirement Phase; E_Re_Des: Review effort in Design Phase; E_Re_Cod: Review effort in Coding Phase

E_Re: Review effort; E_PA: Process Audit effort; E_Test: Testing effort

E_Total: Total effort for the project

Table 2. Defects found in Review Activities

	Def_Re_Req	Def_Re_Des	Def_Re_Cod	Def_Re_Total
V2.5	277	68	NA	345
V2.6	18	12	NA	30
V3.0	73	29	18	120
V3.1	253	6	121	380
V3.1.1	60	36	8	104

Def_Re_Req: Defects found in requirement phase by reviewing

Def_Re_Des: Defects found in design phase by reviewing

Def_Re_Cod: Defects found in coding phase by reviewing

De_Re_Total: Total defects found by reviewing

Table 3. Defects found in Testing Activities

	Def_Intro_Req	Def_Intro_Des	Def_Intro_Cod	Def_Test_Total
V2.5	88	97	478	663
V2.6	38	32	114	184
V3.0	93	296	2095	2484
V3.1	8	6	530	544
V3.1.1	1	7	73	81

Def_Intro_Req: Defects introduced in requirement phase and found by testing

Def_Intro_Des: Defects introduced in design phase and found by testing

Def_Intro_Cod: Defects introduced in coding phase and found by testing

Def_Test_Total: Total defects found by testing

3. RESULTS & PROCESS IMPLICATION

3.1 Correlation between effort percentages of three QA activities and process implication

Since we lack the Process Audit effort from V3.1.1, we do a correlation analysis between the effort percentages of the three quality assurance activities based on the other four versions and the coefficients are shown in Table 4. We can see that the effort percentage of testing is inversely proportional to that of both review and process audit with a highly negative correlation coefficient -0.73 and -0.92 respectively. This inverse trend would be displayed more directly in Figure 1. As the review and process audit effort percentages tends to increase, the percentage of testing tends to decrease. This result might imply that if the organization put more effort on review and process audit, they could save effort in testing to achieve the same quality goal.

Table 4. Correlation between Effort Percentages of Three QA Activities

	E_Re%	E_PA%	E_Test%
E_Re%	1.00		
E_PA%	0.40	1.00	
E_Test%	-0.73	-0.92	1.00

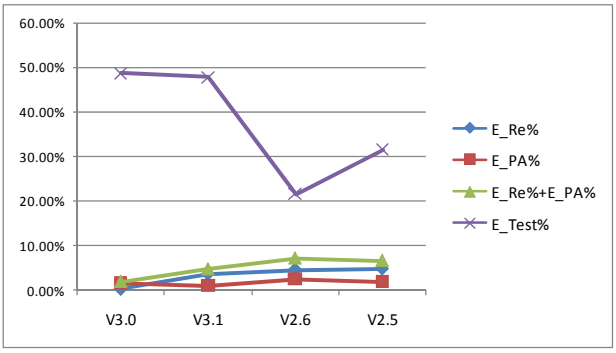

Figure 1. Effort Percentages Trend

To further investigate how much testing effort could save if adding more effort on review and process audit, we do a simple linear regression analysis based on the same data, E_Test% as the dependent variable, E_Re% and E_PA% as the independent variable. And we get the Equation 1 and Table 5 below. Since F value is below 0.05, the regression equation is significant. All other coefficients are significant except the P-value of E_Re% is a little bit above 0.05.

$$E\_Test\% = 0.73 - 2.69 * E\_Re\% - 15.73 * E\_PA\% \qquad \textbf{Equation 1}$$

Table 5. Regression Analysis

	Coefficients	t Stat	P-value
Intercept	0.73	53.82037	0.011827
E_Re%	-2.69	-10.96282	0.057911
E_PA%	-15.73	-19.24302	0.033053
Multiple R	0.99936466		
R Square	0.99872972		
Adjusted R Square	0.99618917		
Significance F	0.03564096		

From this equation, we could see that assuming that other factors are the same if we increase review effort by 1%, we could save testing effort by 2.69%. The same analysis for process audit is that if we increase process audit effort by 1%, we could save testing effort by 15.73%. The ROI of putting more effort on the review and process audit would be (2.69%-1%)/1%=1.69 and (15.73%-1%)/1%=14.73 respectively.

Further considerations about this linear equation would be: First, under the extreme condition that E_Re% and E_PA% are both equal to 0, which means the organization doesn't have any activities related to review and process audit, we could see that the testing would take up as high as 73% of the total effort to satisfy the required quality goal. This percentage of testing is far beyond the benchmark from industry at the same level of quality which implies that the bad effort allocation leads to a huge waste of resources and effort. Second, if the coefficient of E_Re% or E_PA% stands in the range from -1 to 0, which means if you add 1% more effort to review or process audit, it will save less than 1% testing effort, so the ROI of adding more effort on review or process audit will be negative. In such cases, you should reconsider about reducing the effort put on review and process audit, and put more on testing. In sum, this equation gives quality managers some insights about how to balance the effort put on testing, process audit, and review based on history project data.

Process Implication: For this case study, we could see that testing is the most labor-intensive activities of the QA activities and the effort put on review and process audit is insufficient, this might give the quality manager some process improvement implication that putting more effort on review and process audit to save the testing time, and finally shorten the schedule and save the whole cost. Also, since the ROI of adding more effort on process audit is larger than that of adding more effort on review, it also gives some implications that for this software organization, improving the process audit activity might be more cost-effective.

3.2 Cost of finding and removing defects in each phase and process implication

In order to compare cost of finding and removing defects at each phase, we use Equation 2 below based on the data from Table 1, Table 2 and Table 3. Because we lack some data of reviewing effort and defects in coding phase, we only compare the requirement, design and testing phase' s cost of finding and removing defects here, and the result is shown in Table 6 and Figure 2.

Cost of finding and removing defects= Hours spent on reviewing or testing in the phase/ numbers of defects found and removed in the phase ***Equation 2***

We can see that the hours spent on finding and removing per defect is increasing from requirement to testing, this means that the cost of finding and removing defects is increasing as the project goes on. This finding from this Chinese organization is also similar to other surveys carried out by IBM, TRW, GTE and summarized by Boehm [6].

Process Implication: The finding from this analysis continue to support that reviewing is more effective and efficient to detect defects than testing. Adding more effort on early review and defects finding is more cost-effective.

Table 6 . Hours per defect spent on finding and removing defects in each phase

(Hours/defect)	Hrs/Def_Req	Hrs/Def_Des	Hrs/Def_Test
V2.5	0.97	1.94	3.90
V2.6	1.33	1.42	1.02
V3.0	0.57	2.72	6.23
V3.1	1.81	2.17	14.32
V3.1.1	0.58	2.06	4.51
Mean	***1.05***	***2.06***	***5.99***
Median	***0.97***	***2.06***	***4.51***
Stdev	***0.53***	***0.47***	***5.02***

Hrs/Def_Req: Hours spent on finding and removing per defect in requirement phase by reviewing

Hrs/Def_Des: Hours spent on finding and removing per defect in design phase by reviewing

Hrs/Def_Test: Hours spent on finding and removing per defect by testing

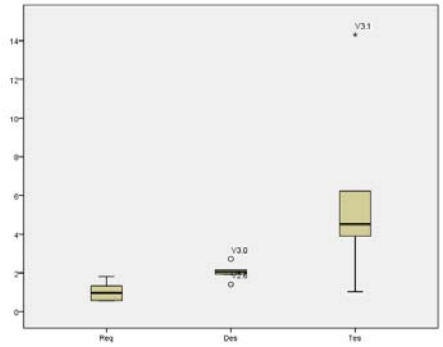

Figure 2. Cost of finding and removing defects in each phase

3.3 Filter Effectiveness of each phase by reviewing-how to improve the reviewing

From the analysis above, we see that it would be better to put more effort on early review. The next question is shall we put equal effort on the requirement review, design review and code walkthrough? A straight answer would be put the extra effort on the weakest part would be more effective. The next analysis will help quality managers to identify which review is the weakest according to the context of the organization.

Review serves as filters, removing a percentage of the defects and let the left to enter to the next phases, the definition the Filter Effectiveness (FE) of each phase:

Filter Effectiveness=defects found in this phase/defects introduced in this phase ***Equation 3***

For simplicity, in our case, we define the Filter Effectiveness as follows:

Filter Effectiveness= defects found in this phase/ (defects found in this phase+ defects introduced in this phase and found by testing)

Equation 4

Process Implication: Based on the data from Table 2 and Table 3, we get the Filter Effectiveness of each phase by reviewing in Table 7. We could see that the Filter Effectiveness decreases as the project goes on, especially in coding phase, as low as 0.01 in V3.0. The process implication here might be that the development

team relies highly on the testing group. They should put more effort on code walkthrough and unit testing by peer review.

Table 7 . Filter Effectiveness of Each Phase by Reviewing

Percentage	FE_Re_Req	FE_Re_Des	FE_Re_Cod
V2.5	0.76	0.41	NA
V2.6	0.32	0.27	NA
V3.0	0.44	0.09	0.01
V3.1	0.97	0.50	0.19
V3.1.1	0.98	0.84	0.10
Mean	***0.69***	***0.42***	***0.10***
Median	***0.76***	***0.41***	***0.10***
Stdev	***0.30***	***0.28***	***0.09***

3.4 Causal analysis for process audit-how to improve process audit

From the analysis in section 3.1, it would be beneficial if we put more effort on process audit too and this might even be more effective than review. The independent auditors check the process with a bunch of checklists for the following process classified as four categories according to CMMI v1.2 [9].

(1). Supporting processes: Major Event Processing, Measurement and Analysis (MA), Quality Assurance (QA), Configuration Management (CM)

(2). Engineering processes: Requirements Development (RD), Requirements Management (REQM), Project Development Lifecycle Standard Process (PDL), Software Testing (TEST), Review (REV), Product Maintenance (PM)

(3). Project management processes: Project Establishment, Project Planning (PP), Project Monitoring and Control (PMC), Risk Management, Project Closing

(4). Process management processes: Organization Training (OT), Software Process Improvement (PIM)

However, the similar question would be asked as above, which process should be put more effort on auditing? And the answer would be the process with the most Not Compatible items (NCs). Due to the data availability of NCs, we could only get the NCs from V3.0, V3.1 and V3.1.1. Based on the three projects, we get a distribution of origins of NCs as shown in Figure 3. We find that the process of Project Monitoring and Control (PMC) and Project Planning (PP) takes most part of the NCs.

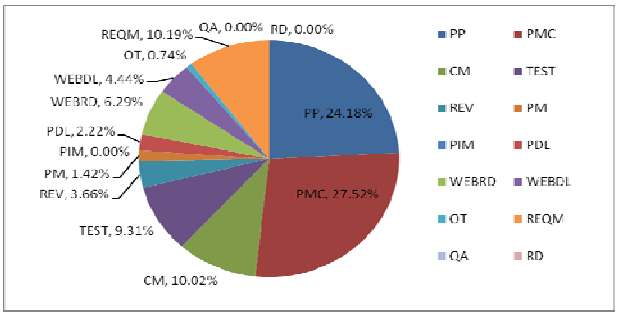

Figure 3. Distribution of Origins of NCs

To get the further information from these NCs, we find that the main problem for PMC is that developers often neglect submitting their periodical reports in time. Main items of these reports include task completion ratio, planned and actual effort, problems or risks they encounter. These reports serve as an important role for the project manager to get to know clearly about each developer's task progress, identify risks, problems, and control and manage the whole project. However, missing reports from developers would blindfold eyes of the project manager, some critical risks or problems might be delayed to solve. For example, if one of the developers forgets to report that he hasn't finished code walkthrough before the project manager establishes a baseline for the testing group to test. You could imagine that how buggy the developer's part would be and this might lead to re-baseline and re-test if some critical defects cause the system crashes down. In such cases, it would be a huge waste of time and cause the quality unsatisfied for the whole project.

By investigating the NCs related to PP, the most common problem is that planned and actual effort is always displaying a huge difference. Overestimating the task effort might lead to project delay and shorten the time for testing and lead to low quality product, and underestimating might lead to resource waste. The large proportion of NCs related to PP shows the weakness of the effort estimation. Further improvement would include adopting more systematic and professional effort estimation methods or tools, such as COCOMO to improve the estimation accuracy.

Process Implication: For this organization, strengthen the process audit on Project Planning and Project Monitoring and Control in the future.

3.5 Distribution of Types of Defects Found by Testing-Testing complements review

From collecting the defects found in testing from V3.0, V3.1 and V3.1.1, we classify the defects according to their types (F-Function, G-Algorithm, U-User Interface, P-Performance) in Figure 4. We find that defects related to function takes the most part and this might imply that the insufficient effort on early review especially insufficient code walkthrough before testing. Defects related to user interface and performance are hard to detect by review but can be more easily detected by testing, so this would verify again that testing activity serves to complement review activity.

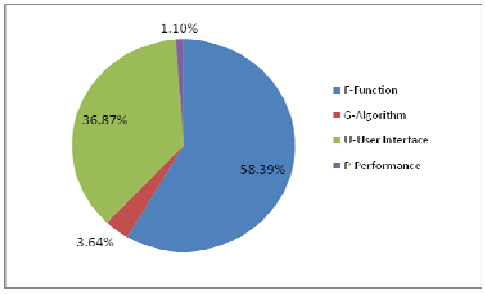

Figure 4. Distribution of Types of Defects Found by Testing

4. CONCLUSION AND FUTURE WORK

In order to identify the most effective effort distribution pattern of QA activities to finally improve the whole process effectiveness, in this study, we do a serials of empirical analysis based on the history project data from a Chinese software organization. The results from this study further verify that early review is more effective than late testing; process audit, at least in this study,

might be even more effective to prevent the defect introduction; these three types of QA activities complement each other to achieve the required quality goal.

The procedure of these analysis would help quality managers to identify which type of QA activity is insufficient while others might be overdone, how to identify and improve the weakest part of QA activities, how to balance the effort allocation and planning for future projects and finally improve the whole process effectiveness under the specific organization context.

We regard this empirical study as the first step towards a more comprehensive quantitative empirical study, where the nature of the quantitative functional forms of the optimal QA effort distribution pattern could be estimated. The empirical study also implies that testing is the most labor-intensive and might be the least effective QA activities. However, testing serves as an irreplaceable role in the software life cycle. Future work would expand the basis of this approach to find a way to balance the effort distribution of QA activities and do more research on value-based testing [10] to improve its effectiveness.

REFERENCES

1. IEEE (1991) "IEEE Std 610.12-1990-IEEE Standard Glossary of Software Engineering Terminology", Corrected Edition, February 1991, in IEEE Software Engineering Standards Collection, The Institute of Electrical and Electronics Engineers, New York.
2. Gilb, Tom, and Dorothy Graham. Software Inspection. Wokingham, England: Addison-Wesley, 1993.
3. Grady, Robert B., and Tom Van Slack. "Key Lessons in Achieving Widespread Inspection Use," IEEE Software, Vol. 11, No. 4 (July 1994), pp. 46-57.
4. Holland, Dick. "Document Inspection as an Agent of Change," Software Quality Professional, Vol. 2, No. 1 (December 1999), pp. 22-33.
5. Humphrey, Watts S. Managing the Software Process Reading, Massachusetts: Addison-Wesley, 1989
6. Boehm, B.W. Chapter 4. Software Engineering Economics. Prentice Hall, (1981)
7. Karl E. Wiegers. Seven Truths about Peer Reviews. Cutter IT Journal, July 2002
8. April A. et al. Process Assurance Audits: Lessons Learned. Proceedings of ICSE 1998, pp.482-485
9. CMMI Product Team. CMMI for Development, Version 1.2. Technical Report CMU/SEI-2006-TR-008, 2006.
10. Li. Q. et al. Bridge the Gap between Software Test Process and Business Value: a Case Study. Proceedings of ICSP 2009, pp. 212-223