5th World Congress for Software Quality – Shanghai, China – November 2011

Quality Improvement by the Real-Time Detection of the Problems --- *DevCast* (Development Forecast) for the Failure Project Prevention ---

Takanori Suzuki
Acroquest Technology Co., Ltd.
Kanagawa, Japan
takanori@acroquest.co.jp

Abstract

In software development, the quantitative management might be actually insufficient because of the influence of the analysis relied on individual.

For example, usual manager of quality assurance judge the level of software quality by the defect density and the defect convergence. However, the testing results is depends on the precision of test items and the skill of testers, etc. Therefore it is difficult to evaluate software quality by the testing results.

For improving such a case, it is necessary that the quality is analyzed in real-time and the projects failure risks are found beforehand.

In this document, I evaluated the effectiveness of multilateral static analysis and the relationship of them with the testing results.

1. Introduction

Only 30% of software development projects can succeed in Japanese software development industry[1]. The success means to manage the software development project within planed quality, cost and delivery. The others, 70% of software development projects cannot finish within the planned criteria. Such situation is seriously abnormal compared with other industries. However, this situation is not only in Japan, but also in other countries and these worse situation are not changed in several years[2].

Quality Assurance and Quality Improvement for software development always be the most important field, but the big improvement has not come yet because of no effective ways in several decades.

Especially, the reasons of failure of recent software development projects are the quality problems. In addition to that, the quality problem affects not only developers and customers, but also the society using the software.

Therefore, I focused on the software quality and propose the method to detect and correct the quality problems in real-time.

2. Invisible quality problem

We usually use metrics to evaluate quality in software test. The representative metrics are defect density (number of defects per volume of codes) and defect convergence (total number of defects per expected number of defect). These metrics are independent from test items, reliability of test items and precision of testing.

We have to consider the following factors for evaluation criteria of test completion.

- Quality of input specification
- Quality of test items
- Quality of defect
- Quality of tester

However, it is difficult to consider these all factors. Therefore, many software products which satisfied criteria in the certain process still include many defects rather than expected.

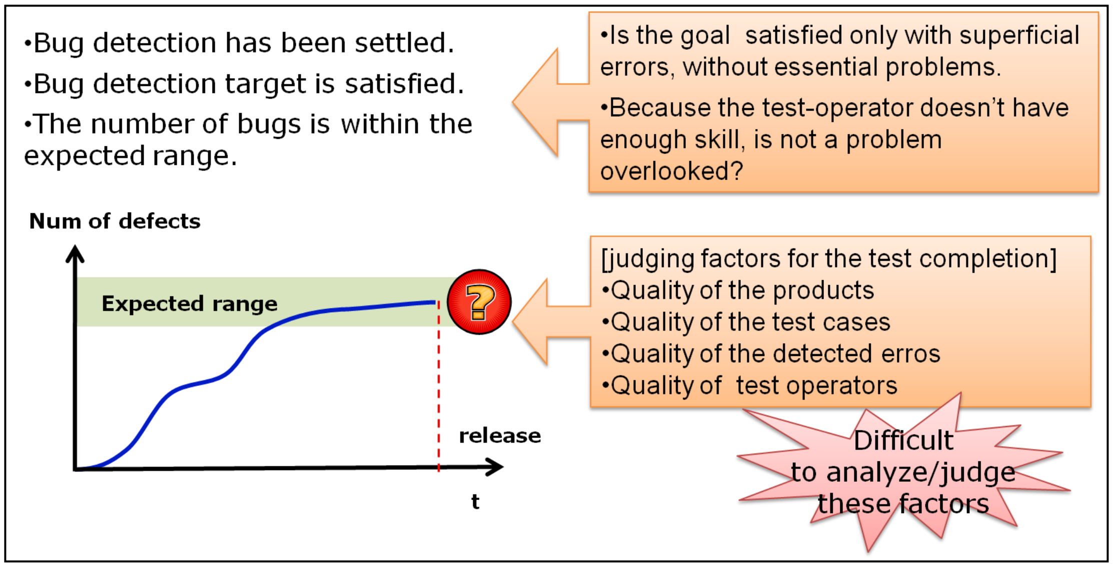
•Bug detection has been settled.
•Bug detection target is satisfied.
•The number of bugs is within the expected range.
•Is the goal satisfied only with superficial errors, without essential problems.
•Because the test-operator doesn't have enough skill, is not a problem overlooked?
Num of defects
Expected range
release
t
[judging factors for the test completion]
•Quality of the products
•Quality of the test cases
•Quality of the detected erros
•Quality of test operators
Difficult to analyze/judge these factors

Figure 1. Test completion and factors for it

QCD(Q:Quality C:Cost D:Delivery) is the basic index for projects management. It is easy to calculate Cost and Delivery. However it is difficult to calculate Quality with simple metrics. That is why it is difficult to detect quality problems and it is almost too late to correct them if you can find out. In actual development site, the priority of quality is usually lower against cost and delivery.

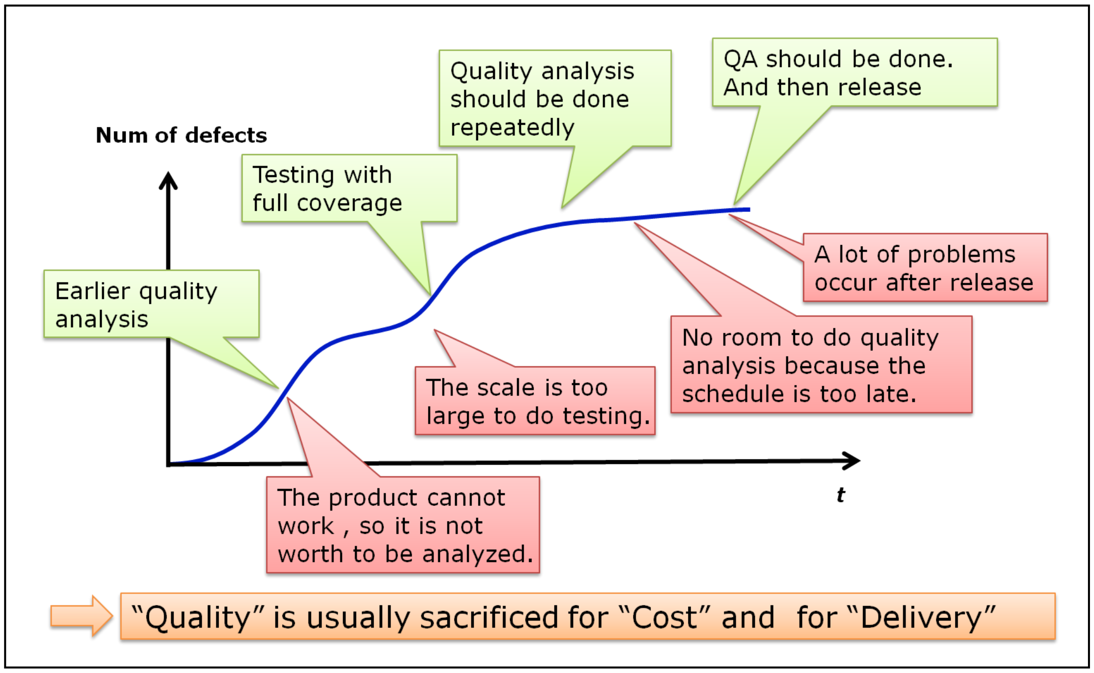
Quality analysis should be done repeatedly
QA should be done. And then release
Num of defects
Testing with full coverage
Earlier quality analysis
A lot of problems occur after release
No room to do quality analysis because the schedule is too late.
The scale is too large to do testing.
t
The product cannot work , so it is not worth to be analyzed.
"Quality" is usually sacrificed for "Cost" and for "Delivery"

Figure 2. The gap between the ideal and the reality in a test process

For breaking through these situations, we need a method with which we judge actual quality of software product in real-time, and predict the future problem and correct them.

3. "Development Forecast" approach

PDCA, cyclic management style, was made popular by Walter A. Schewhart and Dr. W. Edwards Deming, in 1950's[3], and applied for software engineering.

However, In today's software development projects being made to large scale, being shortened for the schedule, being needed to keep rapid-changing technical-elements, every day, being decentralized as off-shore/near-shore.

Then, the complexity and the uncertainty's are just increasing in these projects, and so it is difficult to even make the first plan far from the improvement. And there are a lot of projects fallen through.

Against these situations, I propose an approach, "Development Forecast", which starts with metrics and analysis of the project's situation and which is led to improvement. It is different from the project management based on the planning. It is an approach in which the risk of leading to the failure of the project is in real time feedback, by collecting and analyzing data, based on the fact (product) by the automatic operation or using metrics tools and so on.

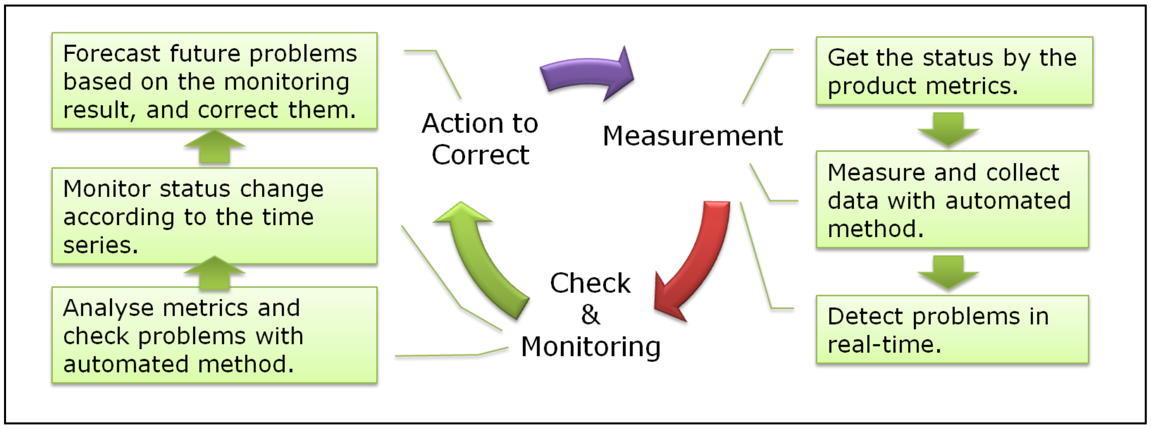
Forecast future problems based on the monitoring result, and correct them.
Monitor status change according to the time series.
Analyse metrics and check problems with automated method.
Action to Correct
Measurement
Check & Monitoring
Get the status by the product metrics.
Measure and collect data with automated method.
Detect problems in real-time.

Figure 3. The approach for the failure project prevention

This time, I applied this approach to a static analysis of the source code, and considered the relativity of the analysis' result and the result of the following test process.

4. Consideration concerning multilateral analysis source code

There is not so much research that evaluates relativity of the static analysis for the source code and the test, though a lot of research showed that static analysis is effective.

In this research, I observed a software development project, using Java language, and evaluated the relativity between the result of the tool based static analysis for source code and the quality evaluation based on the number of the defects in the testing.

4.1 Content of analysis

In this document, I use the metrics in [Table 1. static quality evaluation] :

and I will use the term "static quality evaluation" as these entire evaluation by those two or more viewpoints together.

Table 1. static quality evaluation

No	Metrics	Tool	Description
1)	Coding style violation	Checkstyle	Check source code and count the coding standard violations.
2)	Static analysis violation	FindBugs	Check source code and count the static analysis violations.
3)	Cyclomatic complexity Number	JavaNCSS	Count number of methods having a cyclomatic complexity (McCabe's) greater than 30.
4)	Clone code lines	CPD	Count duplicate code.

The tools[4] used in this research are open-source-tools, generally used on the software development.

In order to compare the results of the analysis for each module, varied in scale, each result -- 1)-3): total value of violations, 4): code-clone lines -- are calculated respectively by dividing on the scale of each module (lines of code).

And then, the effectiveness of the static quality evaluation was evaluated by analyzing relativity between these results of the static quality evaluation and the tendency of the project's defects detected in the integration test and the release test.

4.2 The relativity between the result of static quality evaluation and the result of the software test

There are 5 modules (A - E) in this Project's product, and the results of the static quality evaluation for the modules are showed in the [Figure 4. Approach static quality evaluation results].

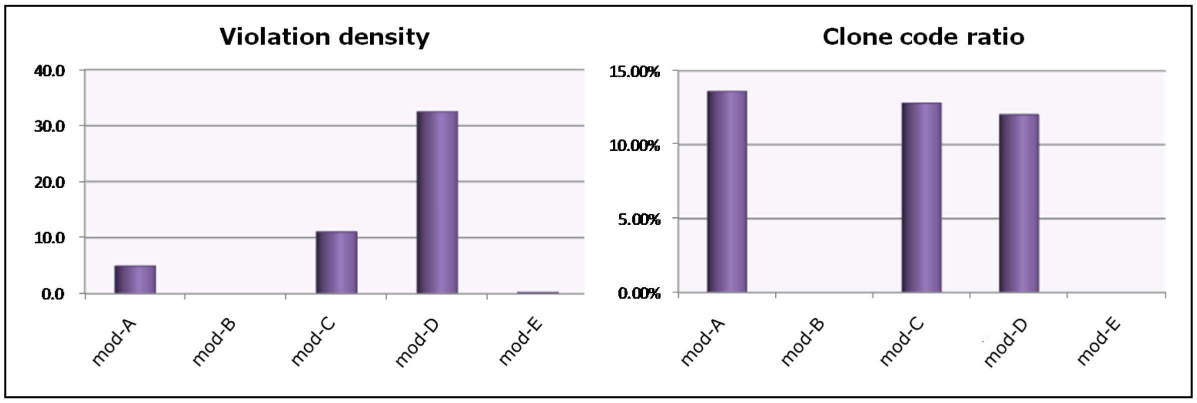

Figure 4. Approach static quality evaluation results

And the results of the integration test and the release test for the modules are showed in the [Figure 5. Defects in the integration test and the release test].

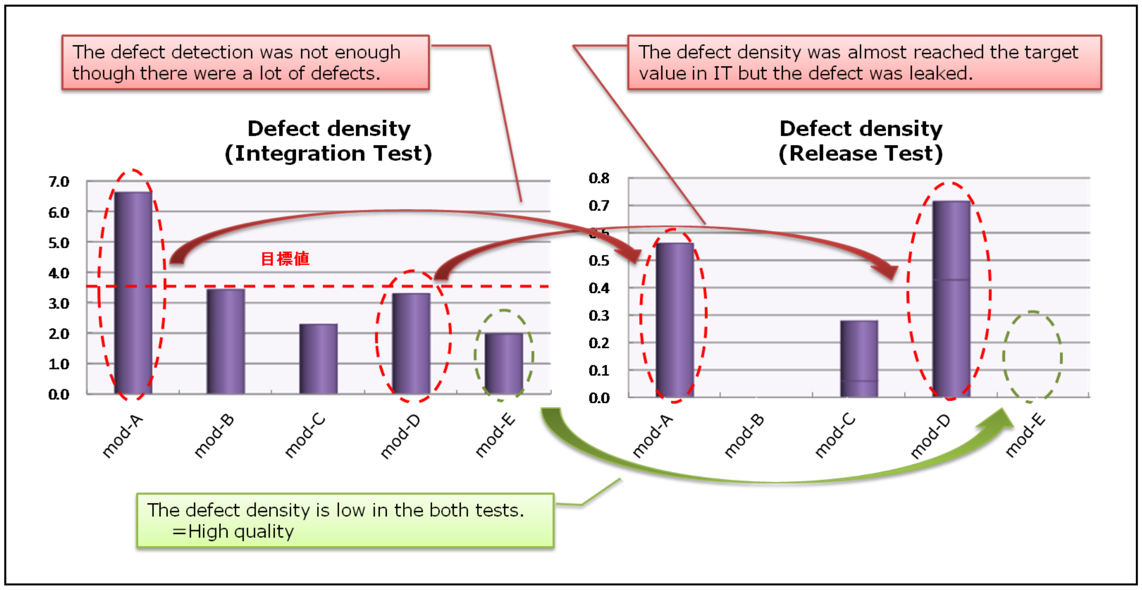

Figure 5. Defects in the integration test and the release test

[Figure 5. Defects in the integration test and the release test] shows the following:

- In a module, a lot of defects were detected in the integration test and the release test (mod-A). In another one, though few defects (not reached the target value) were detected in the integration test, a lot detected in the release test (mod-D).

- Contrary, there was a function with few detection in integration test (not reached the target value) and with few detection in release test, that means the quality is GOOD (mod-C).

According to these result shows how difficult it is to evaluate the quality of the software product only by the test. --- I do not mean to deny the effect of the test, though. ---

On the other hand, seeing the [Figure 4. Approach static quality evaluation results] and [Figure 5. Defects in the integration test and the release test], it is found that many defects were detected by the release test in the module in which many violations are detected through the static quality evaluation(mod-A, C, D).

Then it is thought that the static quality evaluation is one of the efficient measures to specify which module has some quality risks before the test executed, because the tendency of the static quality evaluation and the tendency of the test have the correlation, though the number of defect is not a linear relation.

To be brief, it is expected to improve the software quality before the testing, to specify the modules which contain quality risks by the static quality evaluation for the source codes and to treat the risk properly.

5. Final

In this document, I evaluated the effectiveness of multilateral static analysis, as an approach to prevent project failure as earlier as possible, because the approach will provide us real-time feedback about the risk of failure to project. As a result, it can find the features which have quality risk and improve that.

This quality analysis method can be expected to remove results that relied on individual and to get real-time results because it is based on the automatic analysis of the artifacts by the tools. I think that this approach is effective to prevent failure of the running project.

The following things are thought as future subject.

1. Statistical analysis intended for a lot of projects.

2. Trend analysis of the effects of metric type.

3. Examination and verification of similar analysis in design process.

References

[1] NIKKEI Computer (No.2008-12-1), *2nd project research of 800 companies*, NIKKEI BP, 2008.

[2] Jorge Dominguez, *The Curious Case of the CHAOS Report 2009,* Project Smart, 2009

[3] Wikipedia [Online], http://en.wikipedia.org/wiki/PDCA (accessed on 2008/12)

[4] Checkstyle(http://checkstyle.sourceforge.net/), FindBugs(http://findbugs.sourceforge.net/), JavaNCSS(http://javancss.codehaus.org/), CPD(http://pmd.sourceforge.net/cpd.html)