The Intensity of Fault Slip through in Software Process

Vamsi Mohan V

Abstract — The improvement of software development process is gradually refining. Software companies are trying to deliver bug-free, efficient product in a less time. Now a days, the competition between the IT companies are increased and the companies need improvements that can decrease the lead-time and improve the delivery precision. It is not enough, if companies develop the products on time but also need to be considering the quality of the product. Efficient and effective testing methods find bugs in early stages rather than finding in later stages or in production time. Early catching of bugs will reduce the rework time and also cheaper to find and remove. Rework commonly accounts more than half of the development time. The main reason for high rework cost is because of fault slippage from earlier phases. As an input to this improvement, this article explains the intensity of fault-slippage, which determines the faults that could have been more cost efficient to find in earlier phases of software development and life-cycle.

Index Terms— fault slippage, fault-slip-through, early bug detection, fault latency.

1. INTRODUCTION

Best software processes are being implemented for successful software projects. Even though, good processes implemented in software development there is every possibility of failing projects. Most of the projects are failing to delivery bug-free software and IT industry is doing research and development for delivering successful projects. Software projects will be more stable when most of the faults are removed in earlier phases of development. Hence, project teams should follow different approaches that can reduce the cost of rework.

If more number of bugs exists, rework is commonly needed. In fact, the cost of rework can be reduced by 50% by finding more faults in earlier phases. In some cases proper 'Review & Inspection' (R&I) process also used to reduce the amount of rework. The objective of FST is not only finding the bugs at earlier stages but also finding different types of faults at different phases.

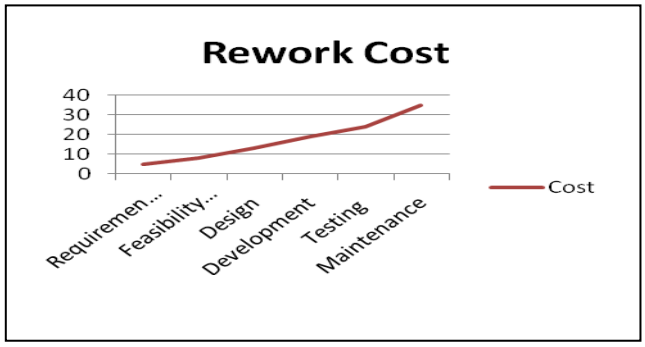
2. FAULT SLIPPAGE MEASUREMENT

Normally the quality level of the product is set at the beginning of the project and project teams try to achieve the defined quality levels (Q-Levels) by following different processes and R&I techniques. The number of faults in software projects has a significant impact on project performance. In order to keep project performance development team and quality teams require giving accurate feedback on the actual quality of the product. They should be transparent in quality reviews and audits. Even in case of outsourced projects, the development team has to give periodical updates to the client on quality levels of the product.

Different kind of measures exist to achieve high quality or to decrease rework cost during development. The number of faults found in early phases is measured as "Fault Slip Through" (FST). Although, FST intended for process assessment, it could also be used for quantifying the benefits of improvement by finding more faults in early stages of software development.

Figure 1. demonstrates, how the cost of faults typically raises in development stage across the software development.

The implication of cost curve is that the identification of software faults early in development cycle is the quickest way to make development more productive. The cost of rework could be reduced up to 30 to 50% by finding more faults earlier.



• ***Vamsi Mohan*** *is a research scholar in Scope management, India. He is an architect and consultant, managed end to end development of multinational projects. An extensive experience in full project lifecycle, software development, application design and architecture, documentation and project planning, database design, systems analysis and database optimization, conversion & integration, technical support; testing and implementation of various strategies in BFSI, CRM and Telecom domains.PH-919901261246. E-mail: vamsi.mohan@yahoo.co.in*

Figure 1. Cost of Rework

Fault Slip Through represents the slippage of faults from one phase to another phase. Due to inefficient fault finding process, faults are slipping from one phase to other phases even though the faults can be finding in a particular phase. Figure 2. show the difference between Fault latency and FST. The intensity of fault slippage will be more, if quality assurance doesn't find the right bug at the right phase.

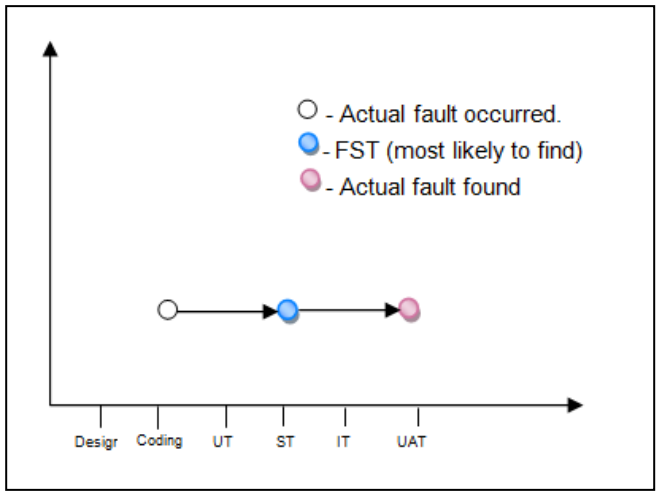


Figure 2. Fault Slippage

The FST is a process to secure that faults are detected in right way in a right phase. This process also implements to enable future efficient work.

Right phase is the phase where fault most likely to find or detect. It is a cost efficient phase to remove and resolve the faults. Early detection of faults and removal process is cheaper than the lateral fault removal. In some particular cases faults might be more cost efficient to detect and fix later in development phases.

The test strategy varies between the organization processes and quality assurance methods, which they follow. But the objective behind FST is to take care of fault slippage from one phase to another phase. i.e., if the organization quality process states that certain types of tests are to be performed at different levels, the FST measure determines to what extent the process adheres to this test strategy. That is, all faults found later than when supposed to find is considered as slippage.
The test strategy or organisation quality process defines in which phase different kind of faults are supposed to be detected. If the feature couldn't test in the earlier phases, those test cases need to be forwarded to next test phases where the feature can be thoroughly tested. That is transparency in testing between different phases.

The purpose of measuring FST is to make sure that the test process finds all possible and right faults in right phase.

The advantages of this process are as follows:

1. Faults slippage will be stopped across the phases.
2. Early bug finding and removal.
3. Reduce in rework cost and time.
4. Similar or repeated faults can be stopped in early phases.
5. Improvement in product quality.
6. Improved delivery precision.

3. FAULT SLIPPAGE IN DEVELOPMENT PROJECTS

The quality standard should set during early phases of software development. The FST can be related to the whole software development process. FST is a process of continuous improvement of the software development.

The FST recommends finding the faults at most cost effective stage through to capture the faults close to the introduction. Each development activity has to be responsible for its own mistakes or errors. With FST measurement, we are assigning responsibility to different teams. Different teams will try to contribute for efficient software development and bug free software delivery.

A Fault Slip Through (FST) process can be divided into below steps:

1. Fault Slip Through analysis and measuring points:
   The fault slip through made for all faults or errors. This analysis is the base for all further measurements.
   To measure FST, five levels has been introduced.
   Fd – Where the fault has detected.
   Fs – Where the fault supposed to be detected.
   Fi – Where the fault has actually introduced.
   TC – Test Case, Was there any test case in Fs phase?

2. Measuring the Fault Slip Through (FST):
   Only Fs and Fd fields shall be used for calculating the Fault Slip Through (FST). There are two ways of measuring FST.
   a) Fault slippage from a phase
   b) Fault slippage to a phase.
   The Quality Assurance team or Line Manage (in case of functional or matrix organization) is responsible to generate Fault Slip Through measurements.

The following data were obtained after applying Fault Slip Through measurement on completed software development project:

Fault Found	Coding	UT	ST	IT	UAT	FST	Total	FST from (in %)
Coding	5	6	24	8	2	40	45	88.9
UT		11	41	21	5	67	78	85.9
ST			17	14	6	20	37	54.1
IT				18	4	4	22	18.2
UAT					0	0	0	0.0
FST	0	6	65	43	17			
Total	5	17	82	61	17			
FST to (in %)	0.0	35.3	79.3	70.5	100.0			

TABLE1. FAULT SLIP THROUGH MATRIX

Not a valid fault Valid fault

Fault Slip Through = FST valid faults * (100/total faults raised)

Table 1. provides an example of FST between different phases. There are many ways to measure Fault Slip Through. FST is not used for performance benchmarking of products and organizations because of the product differences (product maturity, complexity and design variation), process differences (depends on the process model followed by the organization).

3. Quality Management & Improvements:
FST is a concept for continuous improvement and phases with high fault slippage need to be improved. A Root Cause Analysis (RSA) is one way to deal with the problem. It is used to investigate why faults were not detected in the phase where it is supposed (Fs) to detect. Phases with low slippage can be tolerated.

An improvement in FST is important and quality assurance department has to concentrate on this to control rework cost and lead time. Periodical updates are needed to achieve low slippage.

4. CONCLUSION

By following regular FST checking process, the possibility of fault slippage from one phase to another phase will be reduced. This concept will reduce the basic fault slippage. By defining proper roles and responsibilities to the team for measuring FST, this process can be implemented easily. Organizations should take care to implement FST process to reduce lead time and rework cost. This measurement also provides key performance indicators associated to the practise goals. This practise is applied to software development in a specific time frame.

REFERENCES

[1] L-O Damn, L.Lunderberg, C.Wohlin, "Fault-slip-through - A Concept for Measuring the Efficiency of Test Process".

[2] Z. Antolic, FST Measurement Process Implementation in CPP Software Verification, Proceedings MIPRO 2005.

[3] L.-O. Damm, L. Lundberg, C. Wohlin, "Faults-slip-through – A Concept for Measuring the Efficiency of the Test Process", Wiley Software Process: Improvement and Practice, Special Issue, 2006.