Research Paper
Impact Factor: 7.352
Peer Reviewed Monthly Journal
www.ijmdrr.com

IJMDRR
E- ISSN –2395-1885
ISSN -2395-1877

SOFTWARE PROCESS RELIABILITY

Dr. Sattaru Janardhana Rao
Team Lead (Academic Affairs), Andhra Pradesh Information Technology Academy, Vijayawada.

Abstract
The trend of such growth in the telecommunication, business, defense, and transportation industries shows a compound growth rate of ten times every five years. Because of this ever-increasing dependency, software failures can lead to serious, even fatal, consequences in safety-critical systems as well asin normal business. Previous software failures have impaired several high visibility programs and have led to loss of business.

Key Words: Software, Fatal.

Introduction

Software permeates our daily life. A substantial proportion of all products, both commercial products and other application domains, contain some kind of software. The trend is that each product contains more and more software year by year. In particular, science and technology demand high-quality software for making improvements and breakthroughs. The size and complexity of software systems have grown dramatically during the past few decades, and the trend will certainly continue in the future. The data from industry show that the size of the software for various systems and applications has been growing exponentially for the past 40 years. The trend of such growth in the telecommunication, business, defense, and transportation industries shows a compound growth rate of ten times every five years. Because of this ever-increasing dependency, software failures can lead to serious, even fatal, consequences in safety-critical systems as well asin normal business. Previous software failures have impaired several high visibility programs and have led to loss of business.

Software reliability engineering is centered on a key attribute, software reliability, which is defined as the probability of failure-free software operation for a specified period of time in a specified environment. Software reliability is generally accepted as the major factor in software quality since it quantifies software failures, which can make a powerful system inoperative. Software reliability engineering (SRE) is therefore defined as the quantitative study of the operational behavior of software-based systems with respect to user requirements concerning reliability. As a proven technique, SRE has been adopted either as standard or as best current practice by more than 50 organizations in their software projects and reports, including AT&T, Lucent, IBM, NASA, Microsoft, and many others in Europe, Asia, and North America. However, this number is still relatively small compared to the large amount of software producers in the world.

Existing SRE techniques suffer from a number of weaknesses. First of all, current SRE techniques collect the failure data during integration testing or system testing phases. Failure data collected during the late testing phase may be too late for fundamental design changes. Secondly, the failure data collected in the in-house testing maybe limited, and they may not represent failures that would be uncovered under actual operational environment. This is especially true for high-quality software systems which require extensive and wide-ranging testing. The reliability estimation and prediction using the restricted testing data may cause accuracy problems. Thirdly, current SRE techniques or

modeling methods are based on some unrealistic assumptions that make the reliability estimation too optimistic relative to real situations. Of course, the existing software reliability models have had their successes; but every model can find successful cases to just ifyits existence. Without cross industry validation, the modeling exercise may become merely of intellectual interest and would not be widely adopted in industry. Thus, although SRE has been around for a while, credible software reliability techniques are still urgently needed, particularly for modern software systems (Littlewood and Strigini, 2000).

Independently of the field of application, one major factor in focus when using the software is the quality of the software product. With a product quality below expectations, the customers will shortly find a substitute product that better satisfies their needs. Therefore, the software development organizations are forced to assure that their products are of acceptable quality, do not exceed the budget and still would be delivered as per agreed schedule. However, there is still much scope for improvement in the software engineering field. As the software market expands and the customer demands more complex systems, the problems begin to multiply. Thus, in the software engineering discipline, as in many other engineering disciplines, there is much to gain by further research and improvements (Carina Anderson, 2003).

If we take a glance at the history of software, we see that rapid improvement in this sector has taken place since the 1960s. Admittedly, software industry is quite young compared to manufacturing industries. After the industrial revolution, the manufacturing industry at tainted a level of stability, and the idea of continuous process improvement has been accepted throughout the world. In software, however, although the improvement trend is much too steep, organizations still strive for achieving a high level of maturity. Moreover, the characteristics of software make it complex and invisible, making it difficult for the practices in vogue in other industries being applied.

Software Process

The software engineering process is a set of sequential practices that are functionally coherent and reusable for software engineering organization, implementation, and management. It is usually referred to as the software process, or simply the process. Each piece of software is seen as a unique creation. Most programmers would agree that making a program run gives one immense satisfaction.

The objective of every process is to serve the need or request of a customer. Therefore, it is imperative for process definition to work around a customer and his expectation. The customer in this case can be an end customer who consumes the output for the purpose of his business or an intermediate customer who would consume the output in order to carry out another process in the chain of the entire project work. Each process consumes resources such as skills, tools, time and cost, and requires diligent planning and approval.

The quality of the output will depend on the quality of the process, which in turn will depend on the clarity in understanding the expectations of the customer; the quality of input and clarity of the supplier who provides the input. To keep the process evolving, suitable measures should be defined and gathered and controls be applied. Percentage of bugs traceable can be used to measure the efficiency of a review process and suitable training to the reviewers and size of the review team can be the controls applied to enhance the process efficiency. Feedback from each of the components of a process structure is critical to process improvement. (Malik Kamma *et al.*, 2008).

"Software process improvement (SPI) is a systematic procedure to improve the performance of an existing process system by changing the current processes or updating new processes in order to corrector avoid problems identified in the old process system by means of a process assessment."

Software process management is about successfully managing the work process associated with developing, maintaining and supporting software products and software-intensive systems. By successful management, we mean that the products and services produced by the processes conform fully to both internal and external customer requirements and those they meet the business objectives of the organization responsible for producing the products.

At the individual level, the objective of software process management is to ensure that the processes practitioner operate or supervise, meet customer needs, and are continually being improved from the larger, organizational perspective. The objective of process management is to ensure that the same holds true for every process within the organization.

The three key actions needed to establish and maintain control of a software process are to

1. Determine he ther or not the process is under control.
2. Identify performance variations that are caused by process anomalies.
3. Eliminate the sources of assignable causes so as to stabilize the process.

Once the process is under control, sustaining activities must be undertaken to forestall the effects of entropy. Without sustaining activities, processes can easily fall victim to the forces of ad hoc change or disuse and deteriorate to out-of-control states. This requires reinforcing the use of defined processes through continuing management oversight, measurement, benchmarking, and process assessments.

All processes are designed to produce results. The products and services they deliver and the ways in which they deliver them have measurable attributes that can be observed to describe the reliability, quality, quantity, cost, and timeliness of the results produced. If we know the current values of these attributes and if a process is not delivering the qualities we desire,we will have reference points to start from, when introducing and validating process adjustments and improvements (Florac and Carleton, 2007).

References

1. Arnold B.C., And Balakrisha N.,(1989), "Relations, Bounds And approximations for order statistics",Lecturenotes in Statisticsno-53,Springer-Verlag.
2. Balakrishna N., And Cohen A.C.,(1991), " Order Statistics And Inference: Estimation Methods", Academic Press.
3. David,H.A.Andnagaraja,H.N.,(2003)."Order statistics", 3rd edition,Johnwiley& Sons.27-32
4. Kantam R.R.L., And Sriram B., (2001). "Variable Control Charts based Ongamma distribution", Iapqr Transactions 26(2), 63-78.
5. Kantam, R.R.L. And Dharmarao, V., (1994). "Half Logistic Distribution–An improvement overm.L.Estimtor", Proceedings of 11annual Conference of Sds, 39-44.
6. Pham. H., (2006). "System Software Reliability", Springer.
7. Xie,M.,Goh,T.N.Andranjan.P.,(2002)."Some effective Control Chart Procedures For Reliability Monitoring", Reliability engineering and system safety,77,143- 150.