Volume 4, Issue 7, July 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Software Quality Improvement- Cleanroom Approach and Implementation

Mrs. Aparna Sachin Nisal
University of Pune,
Institute of Business Management & Rural Development (IBMRD),
Ahmednagar, Maharashtra, India

Abstract-Quality of a software product is required to be improve or maintain with improvement in process and product standards and with certified reliability, which is an important aspect of every quality software product. The certification process needs a reasonable statistical user testing strategy to measure the software reliability. The cleanroom software development approach assures to minimize or remove defect occurrences with defect prevention by using statistical testing and certification. The software quality metrics such as MTTF, defect density, defect removal efficiency (DRE) and different quality standards such as CMM,six sigma etc. also can be used with the cleanroom development. This cleanroom mechanism reduces testing time as well as effort while performing statistical user testing so that software quality is not diluted and maintain a high degree of software reliability.

Keywords: Cleanroom, CMM, DRE, MTTF, Statistical testing, Reliability, Certification, Quality, Metric.

I. INTRODUCTION

Cleanroom technology is central to high tech manufacturing. Its development stems from the convergence of needs in several fields of manufacturing, health care, and military requirements. Cleanroom software Engineering techniques consist of a body of practical and theoretically sound engineering principles applied to the activity of software engineering. Cleanroom software engineering (CSE) is a managerial and engineering process for the development of high quality software with certified reliability. The combination of CMM management, organizational capabilities, the judicious application and quality conditions for Six Sigma of cleanroom technical practices represents a powerful process improvement paradigm."Do it right the first time"? That's the overriding philosophy of cleanroom software engineering - a process that emphasizes mathematical verification of correctness before program construction commences and certification of software reliability as part of the testing activity.
Two Priorities of cleanroom Software Engineering are,

1) Defect prevention rather than defect removal (any defects not prevented should be removed). This first priority is achieved by using human mathematical verification in place of program debugging to prepare software for system test.
2) To provide valid, statistical certification of the software's quality through representative user testing at the system level. The certification takes into account the growth of reliability achieved during system testing before delivery.

II. TRADITIONAL AND MODERN SOFTWARE TESTING

Many organizations still use such testing methodology (it is also called traditional) where software testing is usually conducted after the build and execution stages.
Traditional (old) way of software testing has the phases as, requirement, design, code & build, testing and maintenance. Unfortunately that is an erroneous methodology because the earlier you find an error – the more funds you can save. But many organizations have improved this way of thinking and choose modern way of software testing. In this modern way testing should carry out in parallel with all traditional development stages. In this testing, after the verification activity of the software product, unit testing, Integration testing, system testing and Acceptance testing are the four levels, which get followed for efficient testing and minimizing the occurrences of defects.

III. SOFTWARE QUALITY METRICS

Software quality metrics focus on the quality aspects of the product, process, and project. They can be grouped into three categories in accordance with the software life cycle: end-product quality metrics, in-process quality metrics, and maintenance quality metrics.
Product quality metrics considers Mean time to failure (MTTF) , Defect density as idea of defect rate and with metrics as Lines of count(LOC) and function points(FP). Also involvement of the customer is the important aspect of quality product.
The process quality metrics considers phase-based defect arrival that is defects related to different phases of SDLC and removal pattern and defect removal efficiency (DRE) as the important quality process activity.

Implementation of many such metrics of software quality help to detect errors, defects and suggest improvements for defect removal and to minimize their occurances.But none of these techniques give assurance of zero defect or defect free softwar.**The cleanroom implementation assures about nearer to zero defect or defect free software with process improvement.**

IV. CLEANROOM SOFTWARE DEVELOPMENT AND PROCESS

A clean room, as defined by modern industrial standards, is a room in which the concentration of airborne particles is controlled, and which is constructed and used in a manner to minimize the introduction, generation, and retention of particles inside the room and in which other relevant parameters, e.g. temperature, humidity, and pressure, are controlled as necessary .In software development a main component of cleanroom is the use of usage based profiles to test the software system. These usage profiles become the basis of statistical tests of the software, resulting in scientific certification of the quality of software system.

Key characteristics of the cleanroom process are an incremental development life cycle and independent quality assessment through statistical testing.

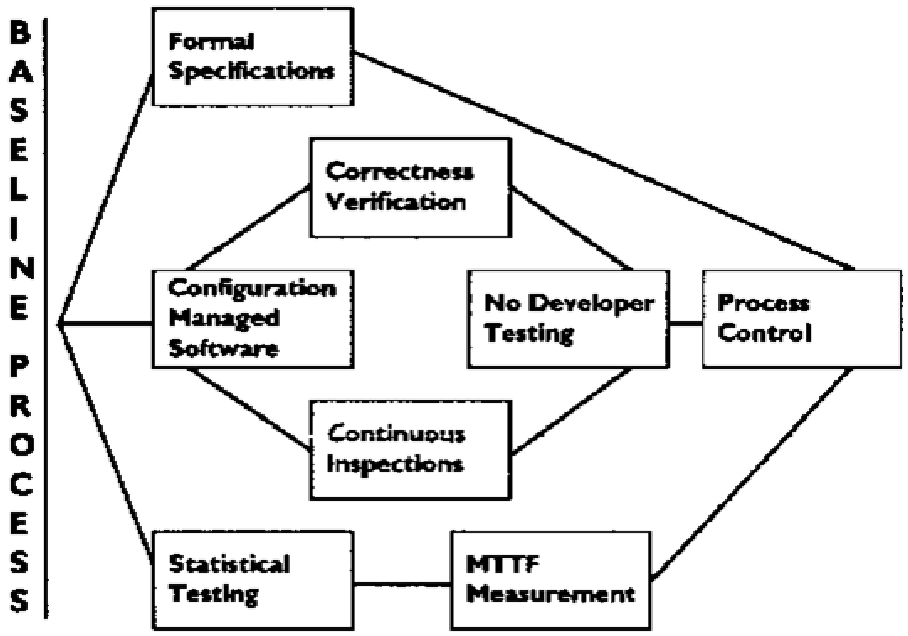

Fig 1 Cleanroom incremental approach with statistical control

The development life cycle starts with a specification that not only defines function and performance requirements, but also identifies operational usage of the software and a nested sequence of user-function subsets that can be developed and tested as increments which accumulate into the final system.

The life cycle of clean room project differs from the traditional life cycle. The traditional 40-20-40 post investigation life cycle consists of 40% design, 20% code and 40% unit testing. Clean room uses 80-20 cycle i.e. 80% for design and 20% for coding. The unexecuted and untested product is then presented to integration testing and expected to work. If it does not work, the defects are examined to determine how the process should be improved. The defective product is then discarded and regenerated using the improved process.

The unit testing does not exist in clean room approach. Due to the combination of formal design methods and mathematical based verification, more than 90% of the total product defects were found before first execution. Drop in the Total Defect Count (by almost half) which highlights the cleanroom focus on error prevention as opposed to error detection.It calls for the development of the software in increments that permit realistic measurements of statistical quality during development, with provision for improving the measured quality by additional testing, by process changes.

The Cleanroom Processes:

The Cleanroom processes are specifically divided as management, specification, development and certification processes.

The Cleanroom Management Processes consist of Project Planning Process, Project Management Process, Performance Improvement Process and Engineering Change Process with the work products as Cleanroom Engineering Guide, Software Development Plan , Project Record, Performance Improvement Plan and Engineering Change Log.

The Cleanroom Specification Processes consist of Requirements Analysis Process, Function Specification Process, Usage Specification Process, Architecture Specification Process and Increment Planning Process with the work products as Software Requirements, Function Specification, Usage Specification, Software Architecture and Increment Construction Plan.

The Cleanroom Development Processes consist of Software Reengineering Process, Increment Design Process and Correctness Verification Process with work products as Reengineering Plan, Reengineered Software, Increment Design and Increment Verification Report.

The Cleanroom Certification Processes consist of Usage Modeling and Test Planning Process and Statistical Testing and Certification Process with the work products as Usage Models, Increment Test Plan, Statistical Test Cases, Executable System, Statistical Testing Report and Increment Certification Report.

The following figure gives above idea of cleanroom processes with process flow information.

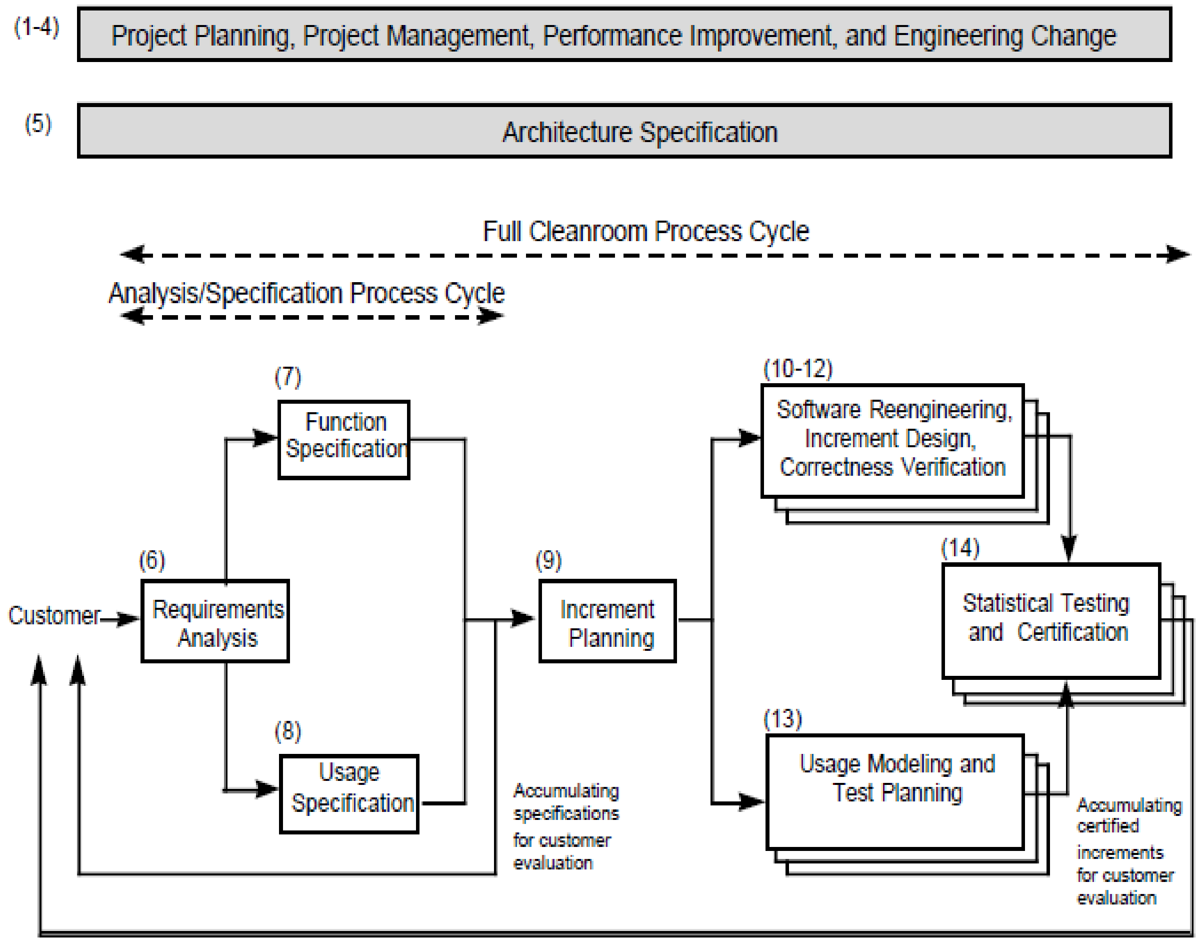
(1-4)
Project Planning, Project Management, Performance Improvement, and Engineering Change
(5)
Architecture Specification
Full Cleanroom Process Cycle
Analysis/Specification Process Cycle
(7)
Function Specification
(6)
Requirements Analysis
Customer
(8)
Usage Specification
(9)
Increment Planning
Accumulating specifications for customer evaluation
(10-12)
Software Reengineering, Increment Design, Correctness Verification
(14)
Statistical Testing and Certification
(13)
Usage Modeling and Test Planning
Accumulating certified increments for customer evaluation

Fig2 Cleanroom process flow

In the cleanroom process, correctness is built in by the development team through formal specification, design, and verification. Team correctness verification takes the place of unit testing and debugging, and software enters system testing directly, with no execution by the development team. All errors are accounted for from first execution on, with no private debugging permitted. Experience shows that **cleanroom software typically enters system testing near zero defects and occasionally at zero defects.** The certification (test) team is not responsible for testing in quality, an impossible task, but rather for certifying the quality of software with respect to its specification. Certification is carried out by statistical usage testing that produces objective assessments of product quality. Errors, if any, found in testing are returned to the development team for correction. If quality is not acceptable, the software is removed from testing and returned to the development team for rework and reverification.The process of cleanroom development and certification is carried out incrementally. Integration is continuous, and system functionality grows with the addition of successive increments. When the final increment is complete, the system is complete.

V. DEFECT PREVENTION AND QUALITY ASSURANCE USING STATISTICAL TESTING, SOFTWARE CERTIFICATION AND CMM

Cleanroom software engineering requires statistical testing by an independent agent for the purpose of certifying software quality. Statistical software testing is a formal process that involves sampling from the intended usage environment and the precise measurement of properties of random variables inherent in such a statistical experiment. The first step in conducting a statistical test is to determine the usage distribution for the software in its intended environment. This distribution is the basis for a random generator of test sequences for the software. In order to achieve this distribution, a usage analysis is performed using the software specification and any available usage information. The usage analysis consists of a top down, structural investigation of the specification document that establishes a set of usage states and defines an ordering on this set.
Cleanroom approach uses Mathematical Verification (MV) to replace program debugging before release to statistical testing. This Mathematical Verification is done by people, based on standard Software Engineering practices which explores that,

1. Human Verification is synergistic with statistical testing- that mathematical fallibility is very different from debugging fallibility.
2. Errors of mathematical fallibility are much easier to discover in statistical testing than are errors of debugging fallibility.

A critical part of the agreement between a producer and receiver is how to measure quality, particularly Statistical Quality. Statistical testing allows the assessment of quality in sigma units. It does not express the defects per KLOC, but gives visibility to the user that how often the defect expected to be encountered.
The set of possible executions of a software system is an infinite population. All testing is really sampling from that infinite population. No testing process, no matter how extensive, can sample more than a minute fraction of all possible executions of a software system. If the sample embodied in a set of test cases is a random sample based on projected usage, valid statistical estimates of software quality and reliability for that usage can be of a software system produces scientific measures of product and process quality.
The objective of the certification team is to provide scientific certification of software fitness for use. The certification team creates usage models which define all possible scenarios of use of the software, together with their probabilities of occurrence. Multiple models can be defined to address different usage environments, or to provide independent certification of stress situations or infrequently used functions with high consequences of failure.

CMM for Quality Improvement:
The Capability Maturity Model for Software (CMM) developed by the Software Engineering Institute, and Cleanroom Software Engineering developed by Dr. Harlan Mills and his associates in IBM and other organizations, share a common concern with software quality and the effectiveness of software development. The principal focus of the CMM is on process management maturity and the principal focus of cleanroom is on rigorous engineering processes. **The CMM management processes and the cleanroom engineering processes are complementary and mutually reinforcing**. The CMM is organized into five maturity levels. The maturity levels are defined in terms of 18 key process areas (KPAs) that characterize project performance at each level. Some of them are Requirements Management, Software quality assurance (SQA), Software product engineering, Software project and configuration management, Quantitative process management, Defect management, change management etc.Each key process area of the CMM is mapped to elements of the cleanroom processes. KPA goals, commitments, abilities to perform, activities, measurements, and verifications are listed, together with the principal cleanroom process implementation and an assessment of the correspondence of cleanroom to the KPA elements.

VI. IMPLEMENTATION OF CLEANROOM APPROACH

Several major products have been developed using cleanroom and delivered to customers. These include:

1. The IBM COBOL Structuring Facility program product was developed using cleanroom software engineering technology in a pipeline of increments with very high quality and productivity. In this cleanroom approach, programs are developed under statistical quality control and mathematical verification. The formal methods of specification, design, functional verification, and testing are described, together with development and management practices required for maintaining intellectual control over the process

2. IBM AOExpert/MVS™ system outage analyzer. (107,000 lines of code) This product combined knowledge-based techniques with systems software. Despite its complexity, it achieved a more than ten-fold reduction in total errors per thousand lines of code found during testing, compared to similar projects. At the same time it reported a three-fold improvement in productivity. No operational errors were reported during beta testing.

3. Ericsson Telecom OS32 operating system. (350,000 lines of C and PLEX) This telecommunications switch operating system reported a rate of one failure per thousand lines of code in testing, a seventy percent improvement in development productivity, and a 100 percent improvement in testing productivity.

Further, An IBM Language Product (40,000 lines of code), An Air Force Contract Helicopter Flight Program (35,000 lines), A NASA Contract Space-Transportation Planning System (45,000 lines) are also experiences of cleanroom implementation. Many of these cases experience that human verification could replace debugging in software development and informal human verification can produce software sufficiently robust to go to system test without debugging. All these projects showed productivity equal to or better than expected for ordinary software development.

VII. CONCLUSION

The cleanroom software engineering process is an evolutionary step in software development. It is evolutionary in eliminating debugging because more and more program design has been developed in design languages that must be verified rather than executed.It is evolutionary in statistical testing because with higher quality programs at the outset, representative-user testing is correspondingly a greater fraction of the total testing effort. Software quality can be engineered under statistical quality control and delivered with better quality. The Cleanroom process gives management an engineering approach to release reliable products with improvement in software quality.

REFERENCES

[1] Rogger R. Pressman,"Software Engineering A Practitioners Approch, Mc Graw Hill International Edition,2010.

[2] Richard C. Linger,"Clean room Software Engineering for Zero- Defect Software", IBM Cleanroom Software Technology Center,1993.

[3] H. Cosmo, E. Johansson, P. Runeson, Sixtensson and C. Wohlin, "Cleanroom Software Engineering in Telecommunication Applications", Proceedings Software Engineering and Its Applications, pp. 369-378, Paris, France, 1993.

[4] Selby, R.W. ,Basili, V.R. and Baker, F.T.,"Cleanroom Software Development: An Empirical Evaluation",Software Engineering, IEEE Transactions,Volume:SE-13 , Issue: 9 ,Sept 1987,Available at: (ieeexplore.ieee.org/Xplore/)

[5] Richard C. Linger,Mark C. Paulk and Carmen J. Trammell ,"Cleanroom Software Engineering Implementation of the Capability Maturity Model (CMM) for Software",Technical Report,Dec 1996,Available at: (http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=12641)

[6] Richard C. Linger and Carmen J. Trammell," Cleanroom Software Engineering Reference",*Carnegie Mellon University*.

[7] Kamaldeep Kaur,"Cleanroom Software Engineering: Towards High-Reliability Software" International Journal of Computer Sci ence & Technology, IJCST Vol. 2, Iss ue 4, Oct . - Dec. 2011

[8] Grant E. Head,"Six-sigma software using cleanroom software engineering techniques", Hewelt Packard Journal, June 1994.

[9] Manas Kumar Yogi ,"A Novel Approach for Cleanroom Software Testing", Science Park Research Journal, Vol-1, Issue-28, 30 January 2014.

[10] *Harian D.Mills , Michael Dyer and Richard C.Linger* ,"CleanRoom Software Engineering" *Information System Institute , IBM Federal Systems Division* , September 1987

[11] Reed Sorensen ,"A Comparison of Software Development Methodologies", Software Technology Support Center, Available at :- http://web.utm.my/

[12] Barry Boehm,Hans Dieter Rombach and Marvin V. Zelkowitz, " Foundations of Empirical Software Engineering: The Legacy of Victor R. Basili, Trace: Tennessee Research and Creative Exchange,,January 2005

[13] O'Regan**,** Gerard, "Cleanroom and Software Reliability", *Mathematical Approaches to Software Quality*, Springer London,2006,pp 176-196.

[14] D.P. Kelly and R.S. Oshana,"Improving software quality using statistical testing techniques, Information and Software Technology ,42 (2000) , pp801–807.

[15] Robert Oshana,"An Industrial Application of Cleanroom Software Engineering -Benefits Through Tailoring",Raytheon TI Systems, IEEE, 1998.