2321-8045 Science Park Research Journal Vol-1, Issue-28, 30th January 2014

Primary Article

A Novel Approach for Cleanroom Software Testing

Manas Kumar Yogi

ABSTRACT

The Cleanroom method of Software Engineering ensures high-quality software with certified reliability, which is an important aspect of every software product. The certification process needs a reasonable statistical user testing strategy to measure the software reliability. We propose a mechanism to reduce testing time as well as effort while performing statistical user testing so that software quality is not diluted as well as maintaining a high degree of software reliability. We also cover a brief history of cleanroom software engineering approach.

Keywords:

Cleanroom, MTTF, User Statistical testing, Reliability,Certification.

Manas Kumar Yogi

From
Sr.Assistant Professor, Computer Science and Engineering Department
Ellenki Engineering College
Siddipet, AndhraPradesh,India

The Article Is Published On January 2014 Issue & Available At
www.scienceparks.in

DOI:10.9780/23218045/1202013/49

1.Introduction:

1.1 Brief History:

The Cleanroom Software Engineering process was developed by Dr. Harlan Mills of IBM's Federal Systems division. Cleanroom Software Engineering and some of its practices were first published in 1981, but the idea did not really surface in major journals until 1986 .Since 1987, IBM as well as a number of other organizations began to apply Cleanroom techniques to their projects. Since then, the process has evolved to keep up with the changing world of software. Design paradigms have moved from strict top-down structured programming to include the likes of object-oriented design. Users of this process have adapted it to coexist with various tools and techniques.

1.2 Approach :

Cleanroom software engineering places importance on mathematical verification of correctness before program implementation starts and certifies software reliability. It was proposed in 1980's but it has not gained popularity due to following reasons. The first reason is that this methodology is too theoretical, too mathematical and too radical to use in real software development. Secondly, it proposes zero units testing for developers. The last reason is due to its rigorous nature of application in all lifecycle phases not applicable for organizations which are operating at low level of process maturity. These all reasons have persistent cultural resistance at its core but still the advantages of Cleanroom software engineering are many. Cleanroom software engineering requires a development cycle of concurrent fabrication as well as certification of product increments that accumulate into system to be delivered .The cleanroom process has been designed to carry out repeated rehearsal of final measurement during software development and to modify the development process, to get desired level of statistical quality. The purpose for the Statistical Testing and Certification process is to demonstrate the software's performance in a formal statistical experiment. The certification goals are established in the Software Measurement Plan and refined in the

Increment Test Plan document. These goals are expressed in terms of software reliability, growth rate, and coverage of the usage. Software undergoes its first execution in this process. Increments are compiled, the system is built, test cases are executed, and the tests are evaluated. Success is determined by the comparison of the software behavior with that present in the Function Specification. Failures found during testing are documented in the Statistical Testing Report. Values of certification measures are compared with the certification goals and decisions are made as to the status of testing. These decisions determine whether or not to continue testing, to stop testing for changes to the software, or to continue on to final software certification. Evaluations and decisions are regarding product quality and process control are documented in the Increment Certification Report.

The Cleanroom Software Engineering strategy:

A pipeline of software increments is developed by small independent software teams and as each increment is certified ,it is integrated into the whole. So, functionality of the system grows with time. Following steps are followed:

Step 1:Increment planning.
Step 2:Information and requirements gathering.
Step 3:Box Structure specification.
Step 4:Formal Design.
Step 5:Correctness verification.
Step 6:Code generation inspection and verification.
Step 7:Statistical test Planning.
Step 8:Statistical User Testing
Step 9: Certification.

2.Proposed Model :

In this paper,we concentrate on the statistical use testing and propose a new model for statistical use testing.Statistical User testing tends to test the software the way users intend to use it and to perform this certification teams(Cleanroom testing teams) must find a usage probability distribution for the software. The blackbox specification for each increment of the software is analysed to define a set of stimuli that cause behavioral change in the software. Based on interactions with prospective users, scenarios are created and a probability of use is assigned to each stimuli. The stimulus is classified with impact on software functionality which may increase or decrease the stability of the software.

We form the following mapping Table:

Table 1: Impact of user action on software functionality

Program Stimulus Level	Probability	Interval
Extreme	40	1-39
Moderate	40	40-79
Stable	15	80-95
Ineffectual	5	96-99

To generate a sequence of usage test cases, we use a random number generator to obtain values between 1-99 values. Consider the following random sequences of

generation:

Table 2: Usage of Test cases

Sequence Number	Random Number Sequence
1	45-71-55-86-98
2	81-31-20-27-7
3	31-87-2-44-99
4	89-55-30-96-53

Selecting the appropriate stimuli based on the distribution interval shown above the following impact of use cases is obtained:
Moderate- Moderate- Moderate- Stable- Ineffectual (Sequence 1)
Stable- Extreme- Extreme- Extreme- Extreme (Sequence 2)
Extreme-Stable-Extreme- Moderate- Ineffectual (Sequence 3)
Stable-Moderate-Extreme-Ineffectual-Moderate (Sequence 4)
The next step is to form the following table with count of number of occurrences of Extreme impact with corresponding sequence in sorted order:

Table 3: Assignment of ranks for extreme behavioral change in software functionality

Sequence Number	Count of "Extreme " occurances	Rank
2	4	1
3	2	2
4	1	3
1	0	4

The next step is execution of the test cases corresponding to the Sequences according to the rank assigned,

i.e. sequence 2,sequence 3,sequence 4,sequence 1.

The main intention to rank the sequences is to save time and effort while statistically judging the functional behavior of the software. The MTTF (Mean Time To Failure) values are recorded while executing the sequences. The sequences having lowest MTTF represents high reliability for the concerned software increment. The subsequent phase of certification includes that increment without worrying further for any types of failures.

Conclusion and Future Work:

We have presented a strategy keeping in view the testing time required to test statistically the functional behavior of the software according to user's viewpoint. The impact levels for stimulus are based on the specific nature of the software according to the needs of application domain. The model is yet to be validated experimentally. As the user viewpoint of the software behavior changes with respect to time the impact level may also change, so we have to consider the degree of certainty while assigning the ranks.Also, we can extend the

proposed model with determination of MTBF (Mean Time Between Failures) instead of MTTF values as MTBF is considered as more reliable measure than MTTF.

References:

[1] .A. Currit, M. Dyer, and H.D. Mills, "Certifying the Reliability of Software," IEEE Trans. Software Eng., Jan. 1986, pp. 3-1.
[2]. R.W. Selby, V.R. Basili, and F.T. Baker," Cleanroom Software Development: An Empirical Evaluation," Tech. Report TR-1415, Computer Science Dept., Univ. of Maryland, College Park, Md., Feb.1985.
[3] Robert Oshana and Frank P. Clyle "Implementing cleanroom Software engineering into a mature CMM-based software organization" Proceedings of the 1997 International Conference on Software Engineering, Boston United States, pp: 572-573, May 1997.
[4] Richard C. Linger "Cleanroom Software engineering for zero-defect software", Proceedings of the 15th international conference on software engineering, Baltimore, MD USA, pp: 1-13, May 1993.
[5] Robert Oshana "Quality Software Via a Cleanroom Methodology",
http://www.embedded.com/97/feat9609.htm
[6] Cleanroom S/W Engg. - Technology and Process by Stacy J. Prowell, Carmen J.Trammell, Richard C. Linger, Jesse H. Poore.
[7] Cleanroom Software Engineering - Reference Model Version 1.0 by Richard C. Linger, Carmen J. Trammell November 1996,
http://www.sei.cmu.edu/pub/documents/96.reports/pdf/tr022.96.pdf