International Journal of Advance Research in Science and Engineering
Vol. No.6, Issue No. 03 , March 2017
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

CLEANROOM SOFTWARE ENGINEERING: A NEW PARADIGM FOR SOFTWARE DEVELOPMENT

Shaveta Dargan

Assistant Professor, Computer Science, Guru Nanak College for Girls, Sri Muktsar Sahib

ABSTRACT

leanRoom Software engineering is a theory based, team oriented discipline intended to create software with high level of reliability and under high statistical quality control. By using formal methods and rigorous inspections, the aim is to have zero defects and failures in the software. So the life cycle, designing methods, formal approaches and statistical testing principles are all different from the traditional way of software development. Here mathematical based methods, statistical verifications, usage based testing are used to certify the software. This paper introduces the CleanRoom software development approach, how box specifications are used, the processes and the models used in the CleanRoom approach are also discussed. At the end, this paper shows the strength and weakness of CleanRoom software development approach and ends with a conclusion and future work.

Keywords: Reliability, Statistical testing, Quality Control, Correctness verification, Box specifications, rigorous inspections

I. INTRODUCTION

The CleanRoom software engineering is such a software development process with the target of designing software with high reliability and under statistical quality control. The concept was given by Harlan Mills at IBM in 1987. The name is based on hardware CleanRoom and from the process which is used to fabricate semiconductor. Its main motive is "Zero Defect Rate". This means the clean room software engineering believes in defect avoidance rather than defect removal.

Traditional software development life cycle relies on 40% design, 20% code and 40% testing whereas the CleanRoom life cycle depends on 80% design and 20% code and no unit testing. The goal is to spend more efforts, more time on the development and finding errors. By deploying increment methods in writing code and verify their correctness before going to testing phase is the key to have success in the implementation.

CleanRoom Software Engineering is defined as: |**An approach which uses formal design methods, requirement methods and statistical testing procedures with no defects and high reliability**.

"**CleanRoon software Engineering is a set of principles and approaches for the software management, specification, designs and testing used for improving the software quality, productivity and reduces cost. This is possible by emphasizing on defect prevention and defect removal."**

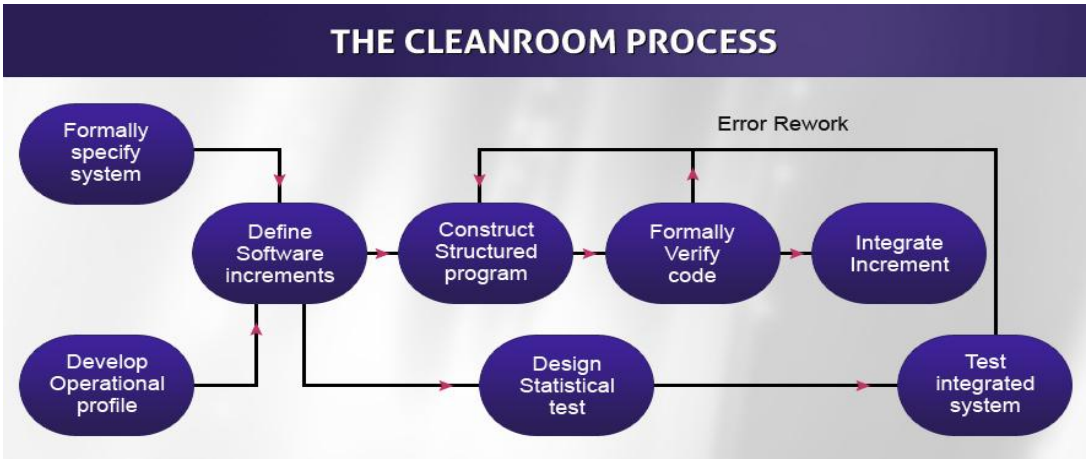
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

It is called **Peer Reviewed Engineering** which uses incremental development methods and disciplined engineering specification and design with full verifications and corrections. The main goal is to build high reliable software with statistical quality certification and control.

II. HOW CLEAN ROOM APPROACH DIFFERS FROM TRADITIONAL APPROACH OF SOFTWARE DEVELOPMENT:

- Various attempts and formal methods are adopted to achieve the target of high reliability and statistical quality whereas there is no guarantee of high reliability in Traditional Approach.
- From specifications, design, coding formal methods is used whereas in traditional development, informal methods are used for specification, design and coding.
- Failures, faults and bugs are not accepted in CleanRoom approach.
- CleanRoom software development is based on incremental approach whereas traditional development is based on the sequential approach.
- Rigid statistical testing methods are used in CleanRoom approach whereas in traditional software development informal methods are used.

III. CLEANROOM SOFTWARE DEVELOPMENT PROCESS



The CleanRoom software development is different from traditional software engineering. The development criterion or the strategy is focused on the following points:-

- **Incremental Planning**: - The project is planned using incremental strategy. Increments are developed, corrected, verified, statistically tested and then certified.
- **Requirements Gathering**: Customers requirements are collected and refined for each increment. The aim is to produce and review informal specifications. The goal of functional specification and design specification is to define the requirements for the software product. The function specification is complete, correct, consistent and traceable only when the requirements are gathered properly.
- **Box structure specification and formal design**: There are three mainly used structures for the specification and design in the CleanRoom software development: Black Box, State Box and the Clear Box

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

specification. CleanRoom software engineering specification and design starts with the external view (Black Box) and is transformed to State Box and is fully developed by the clear box.

- **Correctness Verification**: The CleanRoom software verification team applies a set of rigorous correctness questions and answer by the designing and coding team. Verification begins with the highest level specification and moves towards detailed level of specification.
- **Code Generation, Inspection and Verification**: Box structures specifications are represented in a programming language. Standard walk through and inspections are used to ensure the conformance of code and boxes as well as syntax of the code. Then verification is again done for the code.
- **Statistical test Planning**: Large set of test cases are created to analyze the software. Statistical Testing techniques are used to ensure the quality and reliability of the software.

IV. BOX APPROACH FOR THE FUNCTIONAL SPECIFICATION

All these three structures of clean room approach have identical external behavior but these structures increase the internal visibility to a large extent. Also these structures or the box representations increase data encapsulation and information hiding.

Black Box: A Black box is a specification that explains the external behavior of the function of the system or part of the system. Black Box specification explains that the response is dependent upon the current stimulus and the stimulus history. Clean Room software specification and design begins with the external view i.e. Black Box.

State Box: This is derived from the Black Box specification which uses input data in addition to the stimulus. Here data and services are encapsulated. It is also called the Refinement of the black box specification which specifies the state data required to achieve the black box behavior. The system here produces response and a new state corresponding to the stimuli and previous state.

Clear Box: A clear box specification is again refinement of the state box specification which specifies procedural designs are required to achieve state box behavior and finally low level black boxes are needed to implement the function. Here state transitions are defined explicitly.

ClearRoom Software Engineering begins with the external view (Black Box) and is transformed to State Box and is completed only with the help of procedures given by the Clear Box.

V. REAL WORLD APPLICATIONS OF CLEANROOM SOFTWARE ENGINEERING

CleanRoom software development is an approach for developing software having high statistical quality control. For producing high quality software, CleanRoom approach uses theory based team oriented processes. Using the applications of sound engineering disciplines, cleanRoom approach attempts to produce high quality software. The CleanRoom processes can be used to develop **new software systems** and the **evolution of legacy systems**. The systems can be real-time, embedded, distributed, workstation or client-server.

Here are the applications of CleanRoom Software Engineering:-

- The Systems group at **Raytheon TI Systems**, a SEI level 3 organizations, successfully adopted CleanRoom into a pilot CMM level 5 project.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

- **COBOL Structuring Facility product** (40 KLOC) 1500-3000 errors eliminated in team reviews 740 LOC/Man-Month vs. Industry 150 LOC/Man-Month.
- **NASA Space-Transportation Planning System** (45 KLOC) + productivity 69%, - error rate 45%, - resource usage 60-80%
- **US DoD Picatinny Arsenal project** (90 KLOC) with productivity 320%, 20:1 return-on-investment.
- **Ericsson Telecom OS32 operating system** (350 KLOC), with testing productivity 114%,, design productivity 70%, quality 157% less failures per KLOC 50% and resource usage 4.4%.
- **Flight control**, 33 KLOC, using CleanRoom software engineering in 1987.
- **Commercial products**, 80KLOC.
- **Satellite Control**, 30KLOC, Partial CleanRoom Engineering.
- **Research Projects**, 12KLOC.using CleanRoom software engineering.

VI. COMPARISON WITH OTHER APPROACHES

(A) CleanRoom and capability Maturity Model(CMM):

CMM is basically about management and CleanRoom is fundamentally about the methodology.

CMM has key process areas at level2 that are outside the scope of CleanRoom.

Configuration management and subcontractor management are the important issues which are not addressed by the CleanRoom approaches.

CleanRoom has the focus on the mathematical software development and statistical basis of software testing whereas CMM is silent on the merit of various methods.

(B) CleanRoom and Object Oriented Approach:

Clean Room follows incremental approach while Object Oriented follows iterative development of the project.

Both the approaches use feedback to accommodate the requirements of the user.

Both the approaches use Reuse concept for the software development.

In CleanRoom Box structure decomposition, data objects are created and reused whereas in Object Oriented objects are first identified and ten used in design.

CleanRoom approach uses functional theory and Markov theory to define completeness whereas Object oriented approach uses case representation is typically informal.

VII. WHY CLEANROOM APPROACH IS BETTER

Following are the advantages or most favorable aspects of CleanRoom Software Engineering:-

- Software development based on the increment approach and formal methods.
- Incremental implementation under statistical quality control.
- Sequence based enumerations.
- Peer review engineering.
- Systematic approach to development leads to short development cycles, long product life, zero failures in the field.
- Rigorous specification
- Clarity of the process.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

- Ability to determine exact behavior.
- Proper considerations and steps before coding begin.
- Simplicity.
- Team correctness verification
- Measured Reliability.
- Disciplined engineering specification and design.

VIII. BARRIERS TO CLEANROOM APPROACHES

- Algorithms too theoretical
- Too Radical methods.
- Too mathematical procedures.
- Major difference with correctness verification rather than unit testing.
- Lack of use of defined processes.
- More tools required.
- More Real time issues.
- Mainly used in mature process driven environment

IX. CONCLUSION

CleanRoom approach to software development relies heavily on the statistical Testing and program verification for system reliability certification. The concept is based on Zero Defects present in the software and on the fact that defect detection is more important than defect avoidance. Here designs are simplified, straightforward and verifiable. By using increment development, formal specification, static verification and statistical testing, CleanRoom software development is possible with high reliability and good quality control. So very soon a time will come when traditional approaches and models to software development remains the theoretical concept and in Passive state whereas CleanRoom Approach will remains in Active State with prototyping, object orientation, reuse and verification not testing concepts.

REFERENCES

[1]HTTP://CITESEERX.IST.PSU.EDU/VIEWDOC/DOWNLOAD?DOI=10.1.1.102.1181&REP=REP1&TYPE=PDF

[2] http://www2.latech.edu/~box/ase/papers2011/Cleanroom%20Software%20Engineering.pdf

[3] H. D. Mills, M. Dyer, and R.C. Linger, "CleanRoom Software Engineering," IEEE Software, vol. 4, no. 5, pp. 19-25, Sept. 1987.

[4] E. E. Harrington, “CleanRoom Software Engineering," Department of Software Engineering, University of Wisconsin-Platteville.

[5] R. Selby, V. Basili, and F. T. Baker, “CleanRoom Software Development: An Empirical Evaluation,” IEEE Transactions on Software Engineering, vol. SE-13, no. 9, pp. 1027-1037, Sept. 1987.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

[6] S. W. Sherer, A. Kouchakdjian, P. Arnold, "Experience Using CleanRoom Software Engineering," IEEE Software, May 1996.

[7] R. C. Linger and C. J. Trammel, "CleanRoom Software Engineering Reference Model," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-96-TR-022, Tech. Rep. ESC-TR-96-022, Nov. 1996.

[8] C. M. Wolak, "Taking the Art out of Software Development - An In-Depth Review of CleanRoom Software Engineering," School of Computer and Information Sciences, Nova Southeastern University, Apr. 2001

[9] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.1181&rep=rep1&type=pdf.

[10] http://www.khambatti.com/mujtaba/ArticlesAndPapers/Cleanroom%20Software%20Development.pdf.