Comparative Analysis of Object-Oriented and Cleanroom Software Engineering Techniques
Prof Sharnil Pandya¹ V Raghuvir Rao² Harish Bhutra³ Bhavik Bansal⁴
¹, ², ³, ⁴Department of Computer Science & Engineering
I, ², ³, ⁴ Institute of Technology Nirma University, Gujarat, India.

Abstract—While Traditional Software Engineering cycle consists of analysis, design, code, test, and debug, Cleanroom software engineering emphasizes the need to build quality and reliable software as it is being developed. Life cycle followed by these methods also differ, Cleanroom follows the incremental development while the object-oriented approach follows the iterative development of the project. In Cleanroom Software Engineering Model, Software Quality is achieved by design and verification itself, not through testing as is case in Object Oriented Software Engineering Model. This built in quality can lower the overall cost of the product, and the designs also tends to be more concise and compact than average. In this paper both these models have been Juxtaposed, Analyzed and Reviewed.

Key words: OOA (Object oriented analysis), UML (Unified Modeling Language), OOD (Object oriented Design), UI (User Interface), Cleanroom, Box model, Certification.

I. INTRODUCTION

Software reliability is one of the major software quality Aspects, which quantitatively expresses the continuity of correct service release [5], both Object Oriented Approach and Cleanroom Approach contribute differently to this. In cleanroom software engineering, unit testing and debugging are replaced by correctness verification and statistically based testing [2]. The usage scenario in these models involves the application of “use case” in Object Oriented Approach and the “usage model” in Cleanroom Approach, both these techniques is for characterizing the user's view early in development [3]. Artifacts from these activities facilitate in design and testing phases of the respective model. And the State machine representation for describing the behavior of a design entity is common for both.

Modern Cleanroom practitioners are closer to their mathematical roots than are modern Object Oriented practitioners. Consequently, representation formats differ, with Object Oriented practitioners using graphics and Cleanroom Practitioners using tables or symbolic formalisms [3]. This is because Cleanroom was designed with focus on mathematics-based software engineering for correct software designs and statistics-based software testing for certification of software reliability.

With respect to the life cycle, Cleanroom follows the incremental development while the object-oriented approach follows the iterative development of the project. Both models are intended to provide opportunity for user feedback and to accommodate changing requirements and share a common methodology concerning reuse [2]. Reuse is an explicit objective in both practices. The Object Oriented class and the Cleanroom common service are the units of reuse [3].

In this paper both Cleanroom and Object Oriented Approach have been broadly divided into three phase Analysis, Design and testing respectively. At each phase both models have been Juxtaposed, Analyzed and Reviewed.

II. SOFTWARE ANALYSIS

Object-oriented analysis (OOA) the first technical activity that is performed as part of OO software engineering. Instead of examining a problem using the classic information flow model, OOA introduces a number of new concepts. Object-oriented analysis (OOA) is based upon concepts like objects and attributes, classes and members while in cleanroom we use box structure specification.

In OOA we define the objects that represent the problem to be solved, the manner in which the classes relate to and interact with one another, the inner workings attributes of objects, and the communication mechanisms that allow them to work together. While in cleanroom a “box” encapsulates the system (or some aspect of the system) at some level of detail.

In second phase we define relationships between different objects using UML model in OOA. While in cleanroom, A “box” encapsulates the system or some aspect of the system at some level of detail.

An Object Oriented analysis model encompasses a description of the static as well as dynamic characteristics of classes that describe a system or product. While in functional specification phase of cleanroom we partition a system moving from essential representation at the top to implementation specifics at the bottom [2]. The object relationship model provides an indication of how classes are connected to one another, and the object-behavior model indicates the behavior of individual objects and the overall behavior of the OO system. While cleanroom

III. SOFTWARE DESIGN

Firstly under OOD during analysis modeling, the user model and structural model views are represented. These models tell about the behavior of the system and work as a foundation for the implementation process by identifying and describing the static structural elements of the system [2]. The OOD Process flow is as follow:

UML is organized into two major design activities: system design and object design.

A. System Design

System design in UML represents the software architecture. In it the analysis model is first partitioned into number of subsystems. Then Concurrency between these subsystems is identified and then these subsystems are allocated to processors and tasks [2]. After it a design is developed for
UI. Then a strategy is selected for Data Management. Next is identification of Global resources and control structure to access them. Control mechanism is designed for system including Task Management.

B. Object Design

If system design is considered as a plan then way to execute this plan is specified under object design, i.e. object design is concerned with the detailed design of the objects and their interactions.

1) Object Descriptions

It can be either protocol (Define each message that the object can receive and the related operation that the object performs when it receives the message) or implementation description in which internal details about the data structures that describe the object’s attributes and procedural details that describe operation are provided [2].

2) Designing Algorithms and Data Structures

For OOD an algorithm is created to implement the specification for each operation. A data structure is designed parallel with algorithm design.

3) Program components and Interfaces

Various components of the system should be described and interfaces between objects and the overall structure must be identified [2].

System and object design in UML are extended to consider the design of user interfaces, data management with the system to be built, and task management for the subsystems that have been specified.

4) Task Management

Under this component the characteristics of each task are determined. A Main task and associated objects are determined and other tasks are linked to it.

5) User interface

The actors and their tasks defined in Use-case serves as an input to UI design. From these inputs a command hierarchy is identified. But UI development is not necessary.

6) Data Management

The two areas of significance under data management are: - Management of critical data related to the software and infrastructure creation for storage and retrieval of data [2].

Whereas in case of Cleanroom Software Development Strict structured programming philosophy is used heavily. Top-down approach is incorporated in which higher level black-boxes are decomposed into subsequent state and clear-box.

1) Design Refinement

Each clear-box specification represents the design of a procedure (sub-function) required to accomplish a state box transition. With the clear box, the structured programming constructs and stepwise refinement are used.

2) Design verification

Correctness verification is heart of cleanroom development. In which correct implementation of specification is verified.

At each level of refinement, formal correctness verification is performed. To accomplish this verification, a set of generic correctness conditions are attached with box-structure. Entire system verification can also be done. This verification procedure is conducted by the teams trained for it.

If the Design phase of both methods is compared,

With respect to life-cycle: OOD follows iterative development whereas cleanroom follows incremental development [3].

In OOD potential objects are first identified and then used in system design. But in case of cleanroom box model objects are created and reused according to their need [3].

In OOD the use case representation is largely informal whereas the cleanroom box model describes all possible behavioral scenarios.

In OOD the hierarchy used is Class parent based whereas box based hierarchy is used in cleanroom.

IV. SOFTWARE TESTING

In Object Oriented engineering, Both Analysis and Design models are tested for consistency and correctness before Implementation phase begins which in not the case in Cleanroom. Because in Cleanroom Software Engineering the analysis and design focus on higher quality from beginning itself, Hence the Analysis and Design models are assumed to be impeccable.

While in Object Oriented testing semantic correctness is judged based on the model’s conformance to the real world problem domain. To determine that, it is presented to problem domain experts. Who should examine the class definitions and hierarchy for omissions and ambiguity [2]. To assess consistency, each class and its connections to other classes should be examined. The class-responsibility-collaboration model and an object-relationship diagram can be used to facilitate this activity [2]. Hence each CRC index card is verified.

The strategy and tactics of testing implemented by both are fundamentally different. Object Oriented Software Testing begins with unit testing wherein each component is individually tested, then progress toward integration testing in which cluster of previously tested units are integrated and then tested, and culminates with validation and system testing wherein focus is on user visible actions and output. While Cleanroom Testing is not divided into such phases. Hence Object Oriented approach can identify error if occurred to a particular portion of code, which is not possible in Cleanroom approach. With Cleanroom, the goal is to construct software with no defects during development; hence it would lead to significantly very lower number of errors.

Given high quality code, testers on a Cleanroom project can focus on determining the reliability of the code. (That is estimating how long the code is expected to execute without failure in production.) They can then improve it, if necessary, rather than spend time finding and fixing an indeterminate number of defects keeping fingers crossed that reliability will be satisfactory in production [1].

In Object Oriented testing approaches, set of test cases are derived to uncover design and coding errors. While in cleanroom testing the goal is to validate software
requirements by demonstrating that a statistical sample of use-cases has been executed successfully.

Object Oriented Testing uses Fault Based Testing to design tests that have a high likelihood of uncovering plausible faults. While primary purpose of testing in Cleanroom is certification of reliability, rather than removal of errors. Statistical usage testing is employed to rehearse the software as it will actually be used to make a statistical estimate of the reliability of the software [1].

Statistical use testing “amounts to testing software the way users intend to use it”. To accomplish this, cleanroom testing teams (also called certification teams) must determine a usage probability distribution for the software [2]. Test cases are then generated for each stimulus according to the usage probability distribution. So the more the probability of a stimulus occurring the more is it tested. This ensures that the chances of end user running into errors in significantly low compared to the approach taken by Object Oriented Testing.

Cleanroom test teams certify the reliability of software using Mean Time to Failures (MTTF). Replacing unit testing (Object Oriented Testing) with team verification reviews (Cleanroom Testing), developers are motivated to deliver code for independent testing that is error-free on first execution. Which reduces MTTF increasing software reliability, but this approach can consume more time to form Statistical use testing, compared to Object Oriented Testing approach.

V. CONCLUSION

In conventional software development, errors are accepted as inevitable, each program module should be unit tested (to uncover errors) and then debugged (to remove errors). When the software is finally released, field use uncovers still more defects and another test and debug cycle begins. The rework associated with these activities is costly and time consuming. But cleanroom development works best when practiced by skilled and committed engineers. Reports of the success of the Cleanroom approach in industry have mostly, though not exclusively, come from people already committed to it like IBM. Transferring this approach to other types of software development organizations which may have less committed and less skilled engineers still remains a challenge.

Object-Oriented methods if combined with Cleanroom can lead to a process capable of producing high quality and predictable results that are also reusable. Thus OO A&D approach can be used for front-end domain analysis and Cleanroom can be used for life-cycle application engineering.

REFERENCES