UNIT 2:

CleanRoom Software Engineering

**Introduction:**

* **Clean room software engineering** is a software development approach to producing quality software. It is different from classical [software engineering](https://www.geeksforgeeks.org/software-engineering-introduction-to-software-engineering/) as in classical software engineering QA (Quality Assurance) is the last phase of development that occurs at the completion of all development stages while there is a chance of less reliable and fewer quality products full of bugs, and errors and upset client, etc. But in clean room software engineering, an efficient and good quality software product is delivered to the client as QA (Quality Assurance) is performed each and every phase of software development.
* The cleanroom software engineering follows a quality approach to software development which follows a set of principles and practices for gathering requirements, designing, coding, testing, managing, etc. which not only improves the quality of the product but also increases productivity and reduces development cost. From the beginning of the system development to the completion of system development it emphasizes removing the dependency on the costly processes and preventing defects during development rather than removing the defects.
* The clean room approach was developed by Dr. Harlan Mills of IBM’s Federal Systems Division, and it was released in the year 1981 but got popularity after 1987 when IBM and other organizations started using it.

**Processes of Cleanroom development:**  
Clean room software development approaches consist of four key processes i.e.

1. **Management**  
   It is persistent throughout the whole project lifetime which consists of project mission, schedule, resources, risk analysis, training, configuration management, etc.
2. **Specification**  
   It is considered the first process of each increment which consists of requirement analysis, function specification, usage specification, increment planning, etc.
3. **Development**  
   It is considered the second process of each increment which consists of software reengineering, correctness verification, incremental design, etc.
4. **Certification**  
   It is considered the final process of each increment which consists of usage modeling and test planning, statistical training and certification process, etc.

* Some of the tasks which occur in clean room engineering process:
* Requirements gathering.
* Incremental planning.
* Formal design.
* Correctness verification.
* Code generation and inspection.
* Statical test planning.
* Statistical use testing.
* Certification.

**Box structure in clean room process:**

Box structure is a modeling approach that is used in clean room engineering. A box is like a container that contains details about a system or aspects of a system. All boxes are independent of other boxes to deliver the required information/details. It generally uses three types of boxes i.e.

1. **Black box –**  
   It identifies the behavior of the system.
2. **State box –**  
   It identifies state data or operations.
3. **Clear box –**  
   It identifies the transition function used by the state box.

**Benefits of Clean Room Software Engineering:**

* Delivers high-quality products.
* Increases productivity.
* Reduces development cost.
* Errors are found early.
* Reduces the overall project time.
* Saves resources.
* Clean room software engineering ensures good quality software with certified reliability and for that only it has been incorporated into many new software practices. Still, according to the IT industry experts, it is not very adaptable as it is very theoretical and includes too mathematical to use in the real world. But they consider it as a future technology for the IT industries.

**Following tasks occur in cleanroom engineering:**  
  
**1. Incremental planning**

* In this task, the incremental plan is developed.
* The functionality of each increment, projected size of the increment and the cleanroom development schedule is created.
* The care is to be taken that each increment is certified and integrated in proper time according to the plan.

**2. Requirements gathering**

* Requirement gathering is done using the traditional techniques like analysis, design, code, test and debug.
* A more detailed description of the customer level requirement is developed.

**3. Box structure specification**

* The specification method uses box structure.
* Box structure is used to describe the functional specification.
* The box structure separates and isolate the behaviour, data and procedure in each increment.

**4. Formal design**

* The cleanroom design is a natural specification by using the black box structure approach.
* The specification is called as state boxes and the component level diagram called as the clear boxes.

**5. Correctness verification**

* The cleanroom conducts the exact correctness verification activities on the design and then the code.
* Verification starts with the highest-level testing box structure and then moves toward the design detail and code.
* The first level of correctness takes place by applying a set of 'correcting questions'.
* More mathematical or formal methods are used for verification if correctness does not signify that the specification is correct.

**6. Code generation, inspection and verification**

* The box structure specification is represented in a specialized language and these are translated into the appropriate programming language.
* Use the technical reviews for the syntactic correctness of the code.

**7. Statical test planning**

* Analyzed, planned and designed the projected usages of the software.
* The cleanroom activity is organized in parallel with specification, verification and code generation.

**8. Statistical use testing**

* The exhaustive testing of computer software is impossible. It is compulsory to design limited number of test cases.
* Statistical use technique executes a set of tests derived from a statistical sample in all possible program executions.
* These samples are collected from the users from a targeted population.

**9. Certification**

* After the verification, inspection and correctness of all errors, the increments are certified and ready for integration.

**Cleanroom process model**

* The modeling approach in cleanroom software engineering uses a method called box structure specification.
* A 'box' contains the system or the aspect of the system in detail.
* The information in each box specification is sufficient to define its refinement without depending on the implementation of other boxes.
* The cleanroom process model uses three types of boxes as follows:  
  **1. Black box**
* The black box identifies the behavior of a system.
* The system responds to specific events by applying the set of transition rules.

**2. State box**

* The box consists of state data or operations that are similar to the objects.
* The state box represents the history of the black box i.e., the data contained in the state box must be maintained in all transitions.

**3. Clear box**

* The transition function used by the state box is defined in the clear box.
* It simply states that a clear box includes the procedural design for the state box.

**History:**

The philosophy of the “cleanroom” in hardware fabrication technologies is really quite simple: It is cost-effective and time-effective to establish a fabrication approach that precludes the introduction of product defects. Rather than fabricating a product and then working to remove defects, the cleanroom approach demands the discipline required to eliminate defects in specification and design and then fabricate in a “clean” manner.

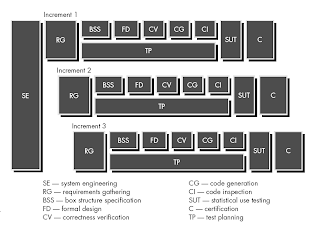
The cleanroom philosophy was first proposed for software engineering by Mills, Dyer, and Linger during the 1980s. Although early experiences with this disciplined approach to software work showed significant promise, it has not gained widespread usage. Henderson suggests three possible reasons:

1. A belief that the cleanroom methodology is too theoretical, too mathematical, and too radical for use in real software development.
2. It advocates no unit testing by developers but instead replaces it with correctness verification and statistical quality control—concepts that represent a major departure from the way most software is developed today.
3. The maturity of the software development industry. The use of cleanroom processes requires rigorous application of defined processes in all life cycle phases. Since most of the industry is still operating at the ad hoc level (as defined by the Software Engineering Institute Capability Maturity Model), the industry has not been ready to apply those techniques.

**The Cleanroom Strategy**

The cleanroom approach makes use of a specialized version of the incremental software model. A “pipeline of software increments” is developed by small independent software engineering teams. As each increment is certified, it is integrated in the whole. Hence, functionality of the system grows with time.

The sequence of cleanroom tasks for each increment is illustrated in figure. Overall system or product requirements are developed using the system engineering methods. Once functionality has been assigned to the software element of the system, the pipeline of cleanroom increments is initiated. The following tasks occur:

[](https://4.bp.blogspot.com/-G44ivhQk4DI/T8H9hMuagYI/AAAAAAAAAgs/7S32Q-v_gkU/s1600/Capture.PNG)

**Increment planning.** A project plan that adopts the incremental strategy is developed. The functionality of each increment, its projected size, and a cleanroom development schedule are created. Special care must be taken to ensure that certified increments will be integrated in a timely manner.

**Requirements gathering.** Using techniques, a more-detailed description of customer-level requirements (for each increment) is developed.

**Box structure specification.** A specification method that makes use of box structures is used to describe the functional specification. Conforming to the operational analysis principles , box structures “isolate and separate the creative definition of behavior, data, and procedures at each level of refinement.”

**Formal design.**Using the box structure approach, cleanroom design is a natural and seamless extension of specification. Although it is possible to make a clear distinction between the two activities, specifications (called black boxes) are iteratively refined (within an increment) to become analogous to architectural and component-level designs (called state boxes and clear boxes, respectively).

**Correctness verification.** The cleanroom team conducts a series of rigorous correctness verification activities on the design and then the code. Verification begins with the highest-level box structure (specification) and moves toward design detail and code. The first level of correctness verification occurs by applying a set of “correctness questions”. If these do not demonstrate that the specification is correct, more formal (mathematical) methods for verification are used.

**Code generation, inspection, and verification**. The box structure specifications, represented in a specialized language, are translated into the appropriate programming language. Standard walkthrough or inspection techniques are then used to ensure semantic conformance of the code and box structures and syntactic correctness of the code. Then correctness verification is conducted for the source code.

**Statistical test planning.**The projected usage of the software is analyzed and a suite of test cases that exercise a “probability distribution” of usage are planned and designed. Referring to figure, this cleanroom activity is conducted in parallel with specification, verification, and code generation.

**Statistical use testing**. Recalling that exhaustive testing of computer software is impossible , it is always necessary to design a finite number of test cases. Statistical use techniques  execute a series of tests derived from a statistical sample of all possible program executions by all users from a targeted population.

**Certification.** Once verification, inspection, and usage testing have been completed (and all errors are corrected), the increment is certified as ready for integration.

Like other software process models discussed elsewhere in this book, the cleanroom process relies heavily on the need to produce high-quality analysis and design models. Box structure notation is simply another way for a software engineer to represent requirements and design. The real distinction of the cleanroom approach is that formal verification is applied to engineering models.

**What Makes Cleanroom Different?**

Cleanroom represents the first practical attempt at putting the software development process under statistical quality control with a well-defined strategy for continuous process improvement. To reach this goal, a cleanroom unique life cycle was defined which focused on mathematics- based software engineering for correct software designs and on statistics-based software testing for certification of software reliability.

Obviously, the cleanroom approach applies most, if not all, of the basic software engineering principles and concepts presented throughout this book. Good analysis and design procedures are essential if high quality is to result. But cleanroom engineering diverges from conventional software practices by deemphasizing (some would say, eliminating) the role of unit testing and debugging and dramatically reducing (or eliminating) the amount of testing performed by the developer of the software.

In conventional software development, errors are accepted as a fact of life. Because errors are deemed to be 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. Worse, it can be degenerative—error correction can (inadvertently) lead to the introduction of still more errors.

In cleanroom software engineering, unit testing and debugging are replaced by correctness verification and statistically based testing. These activities, coupled with the record keeping necessary for continuous improvement, make the cleanroom approach unique.