21.1 The Cleanroom Strategy

Quote:

"The only way for errors to occur in a program is by being put there by the author. No other mechanisms are known. . . . Right practice aims at preventing insertion of errors and, failing that, removing them before testing or any other running of the program."

Harlan Mills

Cleanroom software engineering makes use of a specialized version of the incremental software model introduced in Chapter 2. A "pipeline of software increments" [Lin94b] is developed by small independent software teams. As each increment is certified, it is integrated into the whole. Hence, functionality of the system grows with time.

The sequence of cleanroom tasks for each increment is illustrated in Figure 21.1. Within the pipeline for cleanroom increments, the following tasks occur:

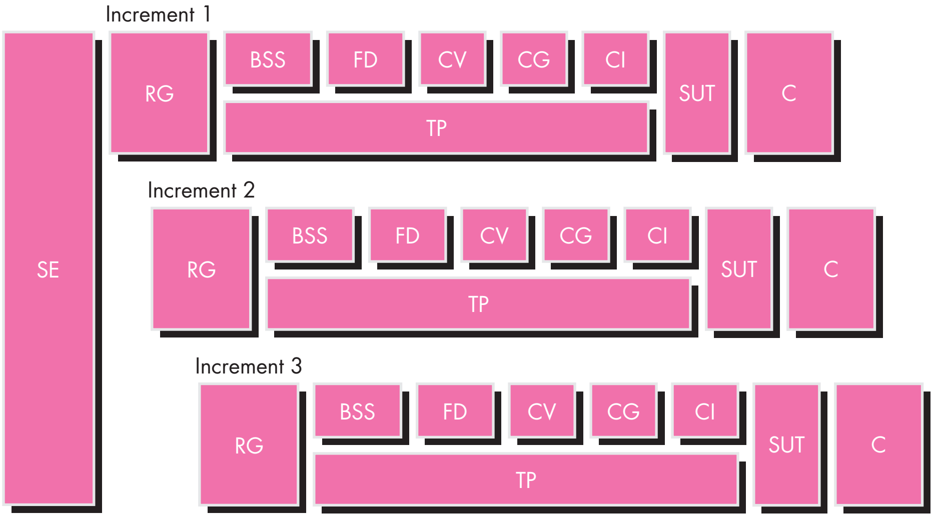
Increment planning. A project plan that adopts the incremental strategy is developed. The functionality of each increment, its projected size, and a cleanroom development schedule are created. Special care must be taken to ensure that certified increments will be integrated in a timely manner.

Requirements gathering. Using techniques similar to those introduced in Chapter 5, 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. Box structures

FIGURE 21.1

The cleanroom process model



SE — system engineering
RG — requirements gathering
BSS — box structure specification
FD — formal design
CV — correctness verification

CG — code generation
CI — code inspection
SUT — statistical use testing
C — certification
TP — test planning

WebRef

An excellent source of information and resources for cleanroom software engineering can be found at **www.cleansoft.com**.

uote:

"Cleanroom software engineering achieves statistical quality control over software development by strictly separating the design process from the testing process in a pipeline of incremental software development."

Harlan Mills

"isolate and separate the creative definition of behavior, data, and procedures at each level of refinement" [Hev93].

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 (Section 21.3.2) 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" [Lin88]. 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. Technical reviews (Chapter 15) 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 is planned and designed (Section 21.4). Referring to Figure 21.1, 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 (Chapter 18), it is always necessary to design a finite number of test cases. Statistical use techniques [Poo88] execute a series of tests derived from a statistical sample (the probability distribution noted earlier) of all possible program executions by all users from a targeted population (Section 21.4).

ADVICE

Cleanroom emphasizes tests that exercise the way software is really used. Use cases provide input to the test planning process.

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

The first four activities in the cleanroom process set the stage for the formal verification activities that follow. For this reason, I begin the discussion of the cleanroom approach with the modeling activities that are essential for formal verification to be applied.