Brady Field

Cleanroom Software Engineering

**Important points**

“As with cleanroom hardware development, the process's first priority is defect prevention rather than defect removal (of course, any defects not prevented should be removed).” [19]

“This first priority is achieved by using human mathematical verification in place of program debugging to prepare software for system test.” [19]

“Its next priority is to provide valid, statistical certification of the software's quality through representative-user testing at the system level.” [19]

“To gain the benefits of quality control during development, Cleanroom software engineering requires a development cycle of concurrent fabrication and certification of product increments that accumulate into the system to be delivered. This lets the fabrication process be altered on the basis of early certification results to achieve the quality desired.” [19]

“Cleanroom software engineering is a practical process to place software development under statistical quality control.” [20]

“Experimental data from projects where both Cleanroom verification and more traditional debugging techniques were used offers evidence that the Cleanroom verified software exhibited higher quality. For the verified software, fewer errors were injected, and these errors were less severe and required less time to find and fix.” [21]

“While it may sound revolutionary at first glance, the Cleanroom software engineering process is an evolutionary step in software development. It is evolutionary in eliminating debugging because, over the past 20 years, more and more program design has been developed in design languages that must be verified rather than executed. So the relative effort for advanced teams in debugging, compared to verifying, is now quite small, even in non-Cleanroom development.” [22]

“The behavior of software is deterministic in that repeating an initial condition and history of use will reproduce the same outputs (with the same failures). But, in fact, if software is used in more than one history by more than one user, the histories of use will usually be different. For that reason, we consider as part of a structured specification a probability distribution of usage histories, typically defined as a stochastic process.” [23]

“In principle, the estimators for software reliability are no more than a sophisticated way to average the times between failure, taking into account the change activity called for during statistical testing. As test data materializes, the reliability can be estimated, even change by change. And with successful corrections, the reliability estimates will improve with further testing, providing objective, quantitative evidence of the achievement of reliability goals.” [24]

**Disagreements**

“The combination of formal design methods and mathematics-based verification had a positive development effect: More than 90 percent of total product defects were found before first execution. This is in marked contrast to the more customary experience of finding 60 percent of product defects before first execution.” [20]

The Authors claim that the Cleanroom method found more than 90 percent of total product defects before the product was ever run. How do they know how many total defects are present in the product? There are always defects present in the code. It would be impossible to measure as a percentage. This comparison is flawed. Perhaps they meant that over 90 percent of the product fixes before shipment were found early using this method. Perhaps they meant that over 90 percent of critical bugs that were found before shipment happened here. They do not say. There is no citation. This measurement is invalid.

**Questions**

I understood everything in the article.