David Lambertson

**Cleanroom Software Engineering Summary**

**Important Points from Article**

* “IBM's Cleanroom process' has uncovered a surprising synergy between mathematical verification and statistical testing of software, as well as a major difference between mathematical fallibility and debugging fallibility in people.” [19]
* “With the Cleanroom process, you can engineer software under statistical quality control. 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]
* “Its next priority is to provide valid statistical certification of the software' quality through representative-user testing at the system level.” [19]
* “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]
* “However, by rethinking the process of statistical quality control itself from a management perspective, we can find a way to put software development under statistical quality control with significant management benefits.” [20]
* “The Cleanroom process permits a sharper structuring of development work between specification, design, and testing, with clearer accountabilities for each part of the process.” [20]
* “This basis requires a software specification and a probability distribution on scenarios of the software's use; it then defines a testing procedure and a prescribed computation from test data results to provide a certified statistical quality of delivered software.” [21]
* “The Cleanroom process has been designed to carry out is principle. It calls for the development are in increments that permit realistic measurements of statistical quality during development, with provision for improving the measured quality by additional testing, by process changes (such as increased inspections and configuration control), or by both methods.” [21]
* “A structured specification is a formal specification (a relation or set of ordered pairs) for a decomposition into a nested set of sub specifications for successive product releases.” [22]
* “Software people customarily talk about errors in the software, typically measured in errors per thousand lines of code. Current post-delivery levels in ordinary software are one to 10 errors per thousand lines. Good methodology produces post-delivery levels under one error per thousand lines.” [23]
* “This objective evidence is itself a basis for management control of the software development to meet reliability goals. For example, process analysis may reveal both unexpected sources of errors (such as poor understanding of the underlying hardware) and appropriate corrections in the process itself for later increments.” [24]
* “Mean time to failure and the rate of change in mean time to failure can be useful decision tools for project management.” [24]

**Things I Didn't Agree With**

“If a program looks hard to verify, it is the program that should be revised, not the verification. The result is high productivity in producing software that requires little or no debugging.” [21]

I disagree with this statement because I do not believe it is possible to have code that requires little or no debugging. Each program should have the same amount of time dedicated to it for debugging, but not every program will produce the same number of errors and failures during said programming.

**Things I Did Not Understand**

I understood the whole article.