1. Important Points
   1. “Studies have shown that for every six new large-scale software systems that are put into operation, two others are cancelled” [63].
   2. “Although the experts could not contrive a road map to guide the industry toward firmer ground, they did coin a name for that distant goal: software engineering” [63].
   3. “If we are ever going to lick this software crisis, we’re going to have to stop this hand-to-mouth, every programmer-builds-everything-from-the-ground-up, preindustrial approach” [63].
   4. “By combining their efforts, academia, industry, and government may be able to hoist software development to the level of an industrial-age engineering discipline within the decade” [64].
   5. “Many fundamental assumptions that programmers make – for instance, their acceptance that everything they produce will have defects – must change in response [to hardware engineers churning out faster, cheaper and smaller machines]” [64].
   6. “The amount of code in most consumer products is doubling every two years” [64].
   7. “Errors in real-time systems such as Clementine are devilishly difficult to spot, like that suspicious sound in your car engine, they often occur only when conditions are just so” [64].
   8. “Software is buckling as well under tectonic stresses imposed by the inexorably growing demand for ‘distributed systems’: programs that run cooperatively on many networked computers” [64].
   9. “To keep up with [an order of magnitude growth in system size every decade,] programmers will have to change the way that they work. ‘You can’t build skyscrapers using carpenters’” [65].
   10. Billions of dollars have been wasted in the funding of failed software [65].
   11. “[Try] to fix not just the bug but also the flaw in the testing process that allowed it to slip through” [66].
   12. “Bugs squashed early rarely threaten a project’s deadline and budget” [66].
   13. “Devastating mistakes are nearly always those in the initial design that slip undetected into the final product” [66].
   14. “Mass-market software producers, because they have no single customer to please, can take a belated and brute force approach to bug removal: they release the faulty product as a ‘beta’ version and let hordes of users dig up the glitches” [66].
   15. “Engineering disciplines share common stages in their evolution” [67]
   16. “Functional tests are still necessary, however, for two reasons” [68]. “First, programmers do occasionally make mistakes in proofs. Secondly, formal methods can guarantee only that software meets its specification, not that it can handle the surprises of the real world” [68].
   17. “I am skeptical that Americans are sufficiently disciplined to apply formals methods in any broad fashion” [68].
   18. “Clean-room development tries to use rigorous engineering techniques to consistently fabricate products that run perfectly the first time” [68].
   19. “Productivity in software development has lagged behind that of more mature disciplines, most notably computer hardware” [68].
   20. “The fact of the matter is that no one really knows how productive software developers are” [68].
   21. “Computer science needs an experimental branch to separate the general results from the accidental” [69].
   22. “If software engineering is to be an experimental science, [sic] that means it needs laboratory science. Where the heck are the laboratories” [69]?
   23. “Programmers have for decades used libraries of subroutines to avoid rewriting the same code over and over. But these components break down when they are moved to a different programming language, computer platform or operating environment” [69].
   24. “[The support of the Indian government for Indian software development] seems to have attracted the attention of large multinational firms eager to reduce both the cost of the software they need and the amount they build in-house” [70].
   25. “Engineers, the infantry of every industrial revolution, do not spontaneously generate. They are trained out of the bad habits developed by the craftsmen that preceded them” [71].
2. Disagreement

“Researches are thus formulating several strategies to attack bugs early or to avoid introducing them at all” [66].

I disagree with the author’s assertion that there could exist a strategy to avoid introducing any bugs into a piece of software. Where there is programming, there will be bugs. Since there will always be programming in the creation of software, software will always have bugs. The only conceivable strategy for avoiding the introduction of bugs at all is to not create the software at all.

1. Questions and Clarifications

I do not have any questions about the content covered by this article.