David Lambertson

**Software Chronic Crisis Summary**

**Important Points from Article**

* “100 computers networked to one another and to 5,000 electric eyes, 400 radio receivers, and 56 bar-code scanners orchestrates the safe and timely arrival of every valise and ski bag.” [63]
* “And some three quarters of all large systems are ‘operating failures’ that either do not function as intended or are not used at all.” [63]
* “Getting software right the first time is hard even for those who care to try.” [64]
* “A major part of the Clementine mission was to test targeting software that could one day be used in a space-based missile defense system.” [64]
* “Errors in real-time systems such as Clementine are devilishly difficult to spot because, like the suspicious sound in your car engine, they often occur only when conditions are just so [see ‘The Risks of Software,’ by Bev Littlewood ICAN, November 1992].” [64]
* “The inconstancy of software development can turn such projects into Russian roulette.” [64]
* “Like handmade muskets, several programs may perform similar functions and yet still be unique in design.” [65]
* “The numbers were unsettling: 55 percent of the projects cost more than expected, 68 percent overran their schedules and 88 percent had to be substantially redesigned.” [65]
* “They assumed that IBM would screen the requirements and design drawn up for the system in order to catch mistakes early, when they can be fixed in hours rather than days. And the FAA conservatively expected to pay about $500 per line of computer code, five times the industry average for well-managed development processes.” [65]
* “The Loral team has learned to control bugs so well, that it can reliably predict how many will be found in each new version of the software.” [66]
* “Yet some bugs inevitably escape detection. The first launch of the space shuttle in 1981 was aborted and delayed for two days because a glitch prevented the five on-board computers from synchronizing properly.” [66]
* “Researchers are thus formulating several strategies to attack bugs early or to avoid introducing them at all.” [66]
* “As a first step, programmers are increasingly stitching together quick prototypes out of standard graphic interface components. Like an architecture’s scale model, a system prototype can help clear up misunderstandings between customer and developer before a logical foundation is poured.” [66]
* 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]
* “Errors were reportedly reduced to just one per 1,000 lines of program code; the industry average is about 25 times higher. Perhaps more important, the company found that development productivity increased by 70 percent, and testing productivity doubled.” [68]
* “In April, MIST announced that it was creating an Advanced Technology Program to help engender a market for component-based software.” [69]
* “The combination of industrial process control, advanced technological tools and interchangeable parts promises to transform not only how programming is done but also who does it.” [71]

**Things I Didn't Agree With**

“A quarter of a century later software engineering remains a term of aspiration. The vast majority of computer code is still handcrafted from raw programming languages by artisans using techniques they neither measure nor are able to repeat consistently.” [63]

I disagree with this statement because in these days, there are so many libraries that we are able to use. These libraries make it so we do not have to keep reinventing the wheel each time. To be more efficient with time and resources, it is quite common to go out and find a library that does what we need and try to use it first so we do not have to handcraft raw code.

**Things I Did Not Understand**

I understood the whole article.