“Silver Bullet” Reading Reaction

The article by Frederick P. Brooks, Jr., was fascinating. Never before had I read such a thorough enumeration of the trials of writing software – and the difficulty, especially, of writing good software. For every task that software tackles admirably, Brooks voiced a major stumbling point in the process of creating software to accomplish these tasks. His arguments spoke from experience, from a deep understanding of the evolving issues of software engineering. He is obviously no newcomer loudly lamenting the failures of the preexisting system, but rather someone who has given the problems enough thought to turn them from frustrations caused by tough-to-pin-down issues felt by innumerable programmers into concise, understandable descriptions of things that are bogging down the software pipeline. His arguments are coherent and immediately applicable even so many years after the article was published.

The idea that there is a “silver bullet” to fix the slow process of software engineering is preposterous. Even in the miniscule projects I have worked upon, the complexity issues alone are enough to bring development to a standstill – and those projects exclude the complexity of the transmission of information between numerous developers. Brooks’ conducts a convincing argument that software development is neither an easy nor straightforward task, and that is one argument that I must agree with wholeheartedly. Brooks’ conclusion that there can inherently be no silver bullet stems from the idea that the complications of software development mostly can be blamed upon the design of the software, rather than the implementation that follows. In this argument, I must agree as well. Modern high-level programming languages afford their users numerous ways of circumventing problems encountered while implementing software, but do not
help a bit with creating an effective organization of program structure. Even 22 years after
Brooks’ writing, there are still no tools that can “do what we want, not what we say,” that is, turn
an idea into an efficient manifestation of a desired product, without regard for any lapses in
judgment by the product’s guiders.

Brooks spends a proportionally large amount of time on the topic of “expert systems”; he
seems somewhat optimistic about their effectiveness. I agree that this focus is warranted, as the
rift between programmers who always use best-practice methodologies and ones who use
whatever method is easiest is huge. However, I do not feel that there is much of a future in a
system that would guide a programmer to use better approaches to the problems he or she is
facing. Oftentimes programmers are rather bull-headed – if they have found success in solving a
programmatical problem in a certain way, they are unlikely to want to deviate from that familiar
(and, after all, successful) method. The “expert system” might help with training beginning
programmers on how to go about solving problems in the best way, such that their practices will
not cause problems later in their career. But I think the required complexity of such systems
creates much of the same problems that “automatic programming” does, in that there is no good
way to tell the computer what you are trying to do (in order for it to guide you along the way),
without actually doing it. The complexity would be far too large, and the effectiveness upon
seasoned programmers far too low, to look toward expert systems as any sort of fix to the
thousands of programmers that seem to be following some worst-practices guide.

In my opinion, Brooks’ quite scientific account of the tribulations of software
engineering should be regarded as a timeless and accurate reference to which we must look to
see how far we have come. If we are able to surmount any of the obstacles that Brooks
explicates, we will know we have done a major service to the practice and to the industry. So
far, however, progress has followed his predictions: slow and minimal, especially compared to the continuing increases in hardware ability. Perhaps one day there will be a tool to intelligently and virtually move from an ill-defined problem to a perfect (or merely serviceable) solution, but I am not holding my breath. I must stick with Brooks, believing that progress will be a hard, slow climb.