No Silver Bullet: A Response

Written by James Yoder

Not many professors have the privilege of having their papers widely published in the software engineering world, but even the papers that do get widely distributed, only a few like Fred Brook’s paper *No Silver Bullet* have become perennial favorites. *No Silver Bullet* has been the cause of controversy and has made software engineers rethink how to improve programming productivity. But 20 years later, developers are still struggling with the same issues that Brooks discussed in his book. Despite advances in programming and new development techniques, no “silver bullet” has been found to solve the ever increasing problem of missed deadlines, over-budget projects, and unusable software.

Up until now, no silver bullets have been found. Object-oriented programming, workstations, extreme programming, and other new software techniques have reduced the time and effort needed to write software. However, none of them have produced the ten-fold increase even in 20 years. Writing software is still essentially hard, albeit many of the classic problems have been pre-solved in libraries. Together, the new designs and techniques can help control the complexity and invisibility problems, but they must all be implemented properly to produce an appreciable increase in productivity. Many of the problems stem from other’s view of software development. Expectations of managers and clients are often unrealistic and are indirectly encouraged by programmers attempting to please their superiors. A system that dramatically increases productivity must give managers a realistic timeframe and amount of functionality while successfully incorporating techniques that can control complexity. Despite the promises of LINQ and MVC making business applications simple, no silver bullet will emerge that can ever make programming easy.

In the past 20 years, the silver bullet has not been found and will not be found in the future. The only hope of increasing productivity is by incremental improvement in techniques and by properly
applying them to software projects. The greatest task today is to solve the social issues that plague software development such as properly implementing the new techniques and reducing the unrealistic expectations that others have about the changeability and difficulty of building software.

**Bibliography**