

Writing Assignment 2

David Houngrinou

CS 390

Dr. Hwang

17 November 2008

In the article *No Silver Bullet: Essence and Accidents of Software Engineering*, Frederick P. Brooks states that there will be no more technologies or "silver bullets" that will help toward programmers' productivity in software engineering. First, this paper will compare the accidental complexity to the essential complexity of software. Next, it will clarify whether the predictions made by Frederick P. Brooks' article are still valid today or not. Finally the text will describe some major technical developments that can be considered as potential silver bullets in the recent years.

The author first presented the complexity of software compared to hardware. The abstraction of software makes it harder to implement. Most developers may agree with this assertion, since the probability of encountering difficulties in writing syntaxes, algorithms and subroutines is very high, even for small applications. The article also differentiated the accidental complexity from the essential complexity of software. The accidental complexity of software can be solved. Generally, it is related to optimization issues, debugging, or fixing human-made errors in the code.

The essential complexity of software is in the problem to be solved. From my student experience in programming, I must admit that software writing is complex. The level of complexity depends on several criteria such as the programming language, the size of the program, or the underlying platform. However, I must criticize the skepticism of the author about silver bullets. There may never be a perfect silver bullet in software writing; but several tools/technologies are now available to simplify the task of developers. Software implementation has become much easier and efficient than it was 10 years ago. As an example, the improvements in object-oriented programming in languages such as C, C++, C# or Java, and principles such as encapsulation or polymorphism can enhance productivity. By emphasizing the

use of objects to design applications, a programmer may have more control on the overall code structure. He can improve modularity of the software, facilitate debugging and make changes to the code without necessarily affecting the entire structure of the program.

I believe that software writing is also complex in the sense that the problem statement is not static and may change during the development process. Generally, clients (companies, individuals ...) submit the software requirements and expect to have a final product released in a certain time frame. During the implementation of the software, the client may add some “slight modifications” to the project requirements. However, there is no “slight modifications” in software. A change in the client requirement may cause the developer to completely discard the current implementation and start over with a different structure. Such situation makes the developer to rethink the problem entirely and spend more time than expected on the same assignment.

The accidental complexity of software has been partially solved with the use of better debuggers and high level languages. However, the essential complexity of software persists. While Frederick P. Brooks claims that there is no silver bullet, he also tends to believe that a series of innovations may lead to significant improvements in software. I doubt there will be a perfect silver bullet; however, several tools are now available to simplify the process of software development. In a close future, I envisage the use of artificial intelligence to solve problems that humans fail to answer.