Unlike computer hardware, the cost of computer software does not rapidly drop and the ability to see twofold gains every two years is extremely difficult if even possible. The idea of a silver bullet for computer software would be something that allows software costs to drop similarly to computer hardware by significantly increasing productivity, reliability, and simplicity. Developing and writing software is essentially hard because software is inherently complex and the ability to achieve the expectations that software needs to conform to different systems and change with customers needs is difficult. Even though there are several techniques and practices that help make developing software easier and possible less costly, there is not an individual technique that could be consider a silver bullet in computer software.

Developing and writing software is essentially hard and will remain difficult as long as the overall general requirements and expectations of software remains the same. Software products are inherently complex as a result of the project being rather large in size and it can be difficult to fully visualize and understand the structure of the software. Most software companies use development teams to produce a product which helps disperse the size of the project onto several people. While there are several different graphical representations that a development team could use to model the software’s structure, it can be difficult to decide which model best represents the software and helps the team member understand the structure of the program. Software products are often expected to conform to various systems and change when the needs of the customers change. This aspect make developing software difficult because technology is continuously changing so the systems that the product is ran on is also changing and changing
software to meet the needs of a customer that was not originally thought of could easily be tricky due to the software’s design. Another main aspect that the article does not go into detail that causes developing software to be hard is communication. Good communication amongst the development team and the client is essential for developing software and can be difficult to achieve at times especially when the team and the client do not see eye to eye on a given aspect of the project.

While there are several techniques and practices that are being used as an attempt to make developing software easier, there was not a single technique that could be considered the silver bullet in the 1990s and there is a small chance that there will be one in the future. Some of the interesting techniques being used to help eliminate accidental problems are object-oriented programming, graphical programming, and program verification. Both object-oriented programming and graphical programming are techniques that developers use to help them more easily understand and organize the software program. Object-oriented programming helps a developer to further break apart a project into several small objects that can be easily understand individually and therefore simplifying the software which allows the productivity of the developer to increase. The problem with object-oriented programming is that it is not the optimal technique to use in every software project and it does not significantly help the productivity of a developer to cut the cost of the project enough to be considered a silver bullet. Program verification can greatly increase the long term reliability of a software product but it does not promise to eliminate the program-testing load nor the cost of labor to verify the program.

The reason why there will probably be no true silver bullet of computer software is that fine tuning the development process to cut the cost of software is a continuous gradual task and can not be do by just one technique. Two strategies that are being used to help reduce the cost of
Bach 3

developing software is buying and modifying the software and rapid prototyping. While buying and then modifying software reduces costs by not having to develop a product completely from scratch, rapid prototyping reduces the cost and helps to satisfy the customer by incrementally producing working models of the program and having the client inspect the product. Both strategies can potentially significantly decrease the cost of developing software but modifying software reduces the significance and impact that the product will have on society and rapid prototyping is a gradual task that still takes time in the development process. The combination of these strategies and the ability to develop great developer for other great developers will help computer software cost to decrease. This combination of strategies and great developers would still not be a silver bullet because it is not just a single strategy that is significantly reducing the cost and each part of the combination is a gradual process.

While developing and writing software is essentially hard, there are several techniques and strategies that can be used to help make the process of developing software a little easier. Since developing software is essentially hard, it is difficult to find a strategy that significantly reduces the cost of computer software and act as a silver bullet in the same way as computer hardware. There are several different strategies that help reduce the cost of software by increasing productivity, reliability, or simplicity but each of these strategies either does not significantly reduce the cost individually or is a process that takes a long time before the significant reduction in cost is seen. There was not a true silver bullet for computer software during the 1990s and even though our ability to develop software is increasing there will probably not be a true silver bullet in the future.