**List of Potential Silver Bullets*:***

1. **Ada and other high-level language advances:** Ada is a general-purpose, high-level language of the 1980s. Its philosophy is more of an advance than the Ada language, the philosophy of modularization, of abstract data types, of hierarchical structuring but it is just another high-level language.
2. **Object-oriented programming:** An order-of-magnitude gain can be made only if the unnecessary underbrush of type specification remaining today in our programming language is itself responsible for nine-tenths of the work involved in designing a program product.
3. **Artificial intelligence:** It’s hard to imagine how image recognition or speech recognition, for example, will make any appreciable difference in programming practice.
4. **Expert systems:** Suggesting interface rules, advising on testing strategies, remembering but-type frequencies, offering optimization hints, etc. It’s difficult finding articulate, self-analytical experts who know why they do things; and developing efficient techniques for extracting what they know and distilling it into rule bases.
5. **“Automatic” programming:** It is the generation of a program for solving a problem from a statement of the problem specifications. The system assessed the parameters, chose from a library of methods of solution, and generated the programs. It is hard to imagine how this breakthrough in generalization could conceivably occur.
6. **Graphical programming:** It has proved to be essentially useless as a design-tool programmers draw flow charts after, not before, writing the programs they describe. No matter what we diagram, we feel only one dimension of the intricately interlocked software elephant.
7. **Program verification:** Program verification does not mean error-proof programs, mathematical proofs also can be faulty. So while verification might reduce the program-testing load, it cannot eliminate it.
8. **Environments and tools:** Language-specific smart editors are developments not yet widely used in practice, but the most they promise is freedom from syntactic errors and simple semantic errors.
9. **Workstations:** The composition and editing of programs and documents is fully supported by today’s speeds. Compiling could stand a boost, but a factor of 10 in machine speed would surely leave think-time the dominant activity in the programmer’s day.