CSCI 362 David Schirduan
Software Engineering 08/29/11

No Silver Response

I enjoyed this description of software development. Brooks addressed many of the problems that we face in this field, and did so accurately while making it easier to understand and very entertaining. However, since this article does pertain to computer science, then it's largest flow is also the only thing which it cannot help: age. This article was written back in 1987. In some ways, the problems that he addresses are even more prevalent now. However, some of his claims are obsolete.

His largest misconception comes from a lack of foresight into the future of high-level languages. For example, he uses Ada his standard for a higher level language. Ada has fallen out of common use for the same criticisms that he gives it. However, several high-level languages have become very popular, and allow for the complexity of programs to be reduced. Python modules are a great example of this. A complex program can call on modules previously made instead of trying to reinvent the wheel. Back in 1987, that sort of access to world-wide modules was not as prevalent as it is today. Also Ada suffers from syntactic difficulty, such as only using a limited set of symbols, with everything else being words and sentences(src: wikipedia). This is inefficient, and has been improved by languages such as python and java by using more mathematical symbols.

The part that was the most prevalent in today's computing world was the difficulty Brooks described with modeling. I loved how he compared laying out a roadway system to designing a program. Programs are not geometrical objects, in fact they are closer to ideas than constructs. Because programs are not geometrical, it is difficult to understand how different areas interact, troubleshoot, and share knowledge with a consistent tool or method. I know from our past class, CSCI360, that it was difficult to model and communicate what we needed to accomplish. Discovering a way to overcome this hurdle would greatly improve larger software projects; which are becoming more and more prevalent as the demand for software and computers grows.

Overall, this was an excellent article. Brooks describes the problem well, and shows that he has struggled with this in his own life. He goes through each possible solution, explains it well, and uses his comparison of the werewolf to keep things interesting and entertaining. Considering the age of this article, and the speed at which computing changes and evolves, it is a wonder that he was so correct in his assessment of the problem. There was no way for him to know of future changes to strongly typed languages, and his knowledge of those high level languages from his time supported his position. Clearly he knew that modeling would be much more difficult to overcome, since it is a problem inherent to programming as a whole. It is interesting that the "silver bullet" of software engineering and design simply lies in a way to successfully communicate among each other.