Frederick P. Brooks, Jr. is considered to be one of the leading authors and contributors to the field of project management and software development. For this discussion, please conduct an online search for articles by and/or about him. Name one of the articles you found and provide its URL, then discuss one of Brooks' contributions and why you feel it is important.

I enjoyed reading about Brooks. His quote, "Adding manpower to a late software project makes it later," also known as "Brooks' Law" reminds me of a project manager's response to management, when it was suggested more software developers would help the project keep it's deadline, the project manager replied, "You can't make a baby in one month with nine women."

I found Brooks' 1987 article, "No Silver Bullet: Essence and Accidents of Software Engineering" very fascinating:

http://www.virtualschool.edu/mon/SoftwareEngineering/BrooksNoSilverBullet.html

In the essay, Brooks reviews a snapshot of the world of emerging technologies, including programming languages, Object-Oriented Programming, Artificial Intelligence, IDEs, Expert Systems, Requirements Gathering, Rapid Prototyping, and Automated Programing, evaluating each of these technologies as a means of overcoming software development's inherent complexity, rapid change (entropy), and invisibility.

Brooks exhibits a great deal of prescience in warning programmers that the complexity of developing software will only be weeded out incrementally. He correctly praises the Ada programming language for reducing complexity, but emphasizes that its philosophy of reducing complexity and emphasis on modularity is the true breakthrough. To this same end, he has the foresight to recognize Object-Oriented programming as another step in the right direction for, "allowing the designer to express the essence of the design without having to express large amounts of syntactic material that add no information content."

Where I disagree with Brooks, is his statement, "Software is invisible and unvisualizable." ER Diagrams, business process modeling, and applications modeling are all pieces of the puzzle toward visualizing software—but perhaps these were simply developments he did not foresee, or did not feel worked well enough to overcome software's invisibility.

At the core of Brooks' thesis is that no one breakthrough or "magic bullet" will overcome software development's complexity; therefore, as we see in the myriad of strategies, methods, and steps a systems analyst must observe in the project lifecycle, only through incremental developments will the complexities in software engineering be mitigated.

What's really impressive about Brooks is his honors and awards, which includes memberships in the NAS, Royal Academy of Engineering, and IEEE, plus awards from the ACM, IEEE, and other Computing and Engineering groups.