Brady Field

No Silver Bullet: Essence and Accidents of Software Engineering

**Important points**

“All software construction involves essential tasks, the fashioning of the complex conceptual structures that compose the abstract software entity, and accidental tasks, the representation of these abstract entities in programming languages and the mapping of these onto machine languages within space and speed constraints.” [180]

“There is no single development, in either technology or management technique, which by itself promises even one order of magnitude improvement in productivity, in reliability, in simplicity.” [181]

“The essence of a software entity is a construct of interlocking concepts: data sets, relationships among data items, algorithms, and invocations of functions.” [182]

“Many of the classical problems of developing software products derive from this essential complexity and its nonlinear increases with size.” [183]

“No such faith comforts the software engineer. Much of the complexity he must master is arbitrary complexity, forced without rhyme or reason by the many human institutions and systems to which his interfaces must conform.” [184]

“The reality of software is not inherently embedded in space. Hence it has no ready geometric representation in the way that land has maps, silicon chips have diagrams, computers have connectivity schematics.” [185]

**Disagreements**

“All successful software gets changed. Two processes are at work. As a software product is found to be useful, people try it in new cases at the edge of, or beyond, the original domain.” [185]

This statement partially depends on how you define successful software, but I would argue that we use software that is still running in its original form in FORTRAN in critical portions of different systems. This is software that is at least 40 years old in many cases. I would definitely consider this software as a success that has not been changed.

**Questions**

I understood everything in the article.